UNIVERSITAT DE LLEIDA

ESCOLA POLITÈCNICA SUPERIOR

ENGINYERIA TÈCNICA EN INFORMÀTICA DE GESTIÓ

Treball de Final de Carrera

Desenvolupament d'una aplicació mòbil en Android per a una botiga de menjar preparat

Autora: Ana Arnalot Baiges

Directora: Montserrat Sendín Veloso

Setembre 2014

Agraïments

A la Montse Sendín per la seva ajuda com a tutora del projecte.

A la meva família pel seu suport.

Al Martí per aguantar-me i animar-me durant tota la duració del projecte i per fer de provador de l'aplicació.

A la Montse, la Natalia, i el Xavi per fer de provadors de l'aplicació.

A l'Assumpta pel logo genial que ha dissenyat.

Al Salva i l'Evelin per muntar el negoci que ha proporcionat la idea de l'aplicació.

A tots els que m'han permès arribar fins aquí.

Índex

Índex de Figures

Índex de taules

Capítol 1: Introducció

1.1. Context del projecte

El tema del projecte va sorgir arran de la problemàtica que els va sorgir a uns amics que van muntar un negoci de menjars preparats. Per augmentar la clientela i donar un valor afegit i diferenciat al seu negoci, va sorgir la idea de muntar un app perquè els clients poguessin consultar els plats disponibles i fer una comanda d'allò què volien aquell dia, i un cop estigués llesta la comanda, els arribés una notificació al mòbil. D'aquesta manera no els caldria estar esperant a la botiga fins que la seva comanda estigués preparada. La idea seria millorar l'eficiència en l'atenció al client i la preparació de comandes.

1.2. Motivació

La motivació del projecte és facilitar el contacte entre el client i el venedor per facilitar la gestió del temps i evitar les esperes innecessàries. A la vegada, permetre al venedor la millor gestió del seu temps i eficiència en la preparació de les comanda, així com facilitar el control de les comandes que hi ha realitzades i quan ja estan preparades per entregar.

En el cas dels clients, tant si és per problemes de temps, com per exemple persones que treballen i tenen poc temps per dinar, com si les comandes són més importants perquè una empresa encarrega un dinar d'empresa o similar, perquè la comanda estigui a punt en el horari establert i ningú tingui problemes d'espera.

1.3. Objectius

Els objectius que es volien aconseguir amb aquest projecte són els següents:

1. Desenvolupar un prototip software d'una aplicació mòbil.

2. Presa de contacte i aprofundiment de les tecnologies emprades en el desenvolupament de l'aplicació mòbil i que s'explicaran amb profunditat més endavant.

3. Treballar amb un servidor web, on emmagatzemar la base de dades i fer peticions sobre ella a través d'un servei web, utilitzant com a motor de base de dades *MySQL*, servidor *Apache* i *PHP* per realitzar les peticions.

1.3.1. Aplicacions existents

Les aplicacions existents amb una funcionalitat similar a la desenvolupada, les que he trobat són les següents, amb la principal diferència que es basen en una sèrie de llocs que ofereixen menjar per emportar i no únicament una botiga.

- *JUST EAT*: de cada botiga que ofereixen apareix un llistat de plats o especialitats i el preu, a partir d'aquí pots fer una comanda. La comanda et permet que ho passis a buscar pel restaurant o bé t'ho porten a casa. La web on es pot trobar mes dades és https://play.google.com/store/apps/details?id=com.justeat.app.es&hl=es.

- *La Nevera Roja*: és molt similar a l'aplicació anterior: a partir de la teva ubicació et mostra els restaurants més propers i pots realitzar una comanda, aquesta indica quina comanda mínima hi ha, i quin temps estimat tarda el restaurant. Es pot trobar més informació en la web https://play.google.com/store/apps/details?id=com.laneveraroja&hl=es.

- *IFood*: en aquest cas a partir de la localització et mostra els restaurants també però al seleccionar un plat et mostra la fitxa. I et permet pagar la comanda. Es pot trobar més informació a la pàgina web de *Google Play* https://play.google.com/store/apps/details?id=br.com.brainweb.ifood&hl=es.

Per fer el disseny de l'aplicació a part de les indicades amb característiques similars també m'he basat en altres tipus d'aplicacions com són les següents:

- *Capraboacasa*: aplicació per fer la compra del supermercat des de el dispositiu mòbil. Es pot trobar informació en la següent pàgina web https://play.google.com/store/apps/details?id=com.caprabo.capraboacasa&hl=es.

- *Amazon*: Es una aplicació on es pot comprar pràcticament qualsevol tipus de producte. Es pot trobar informació en la següent pàgina web https://play.google.com/store/apps/details?id=uk.amazon.mShop.android&hl=es.

- *Recetas de Eroski Consumer*: Es una web on es presenten receptes i es veuen els plats i les característiques. Es pot trobar informació en la següent pàgina web https://play.google.com/store/apps/details?id=es.eroski.recetas&hl=es.

1.4. Fases del desenvolupament

El desenvolupament d'aquest projecte s'ha dividit en diferents fases:

- Decidir projecte a realitzar.
- Estudi de l'arquitectura i funcionament de la plataforma *Android* i de tots els conceptes relacionats per poder desenvolupar l'aplicació
- Anàlisis de les funcionalitats de l'aplicació i estudi dels requisits.
- Disseny de l'arquitectura i anàlisis dels components necessaris per realitzar-la
- Implementar l'aplicació.
- Bateria de proves
- Elaboració de la memòria.
- Elaboració de la presentació.

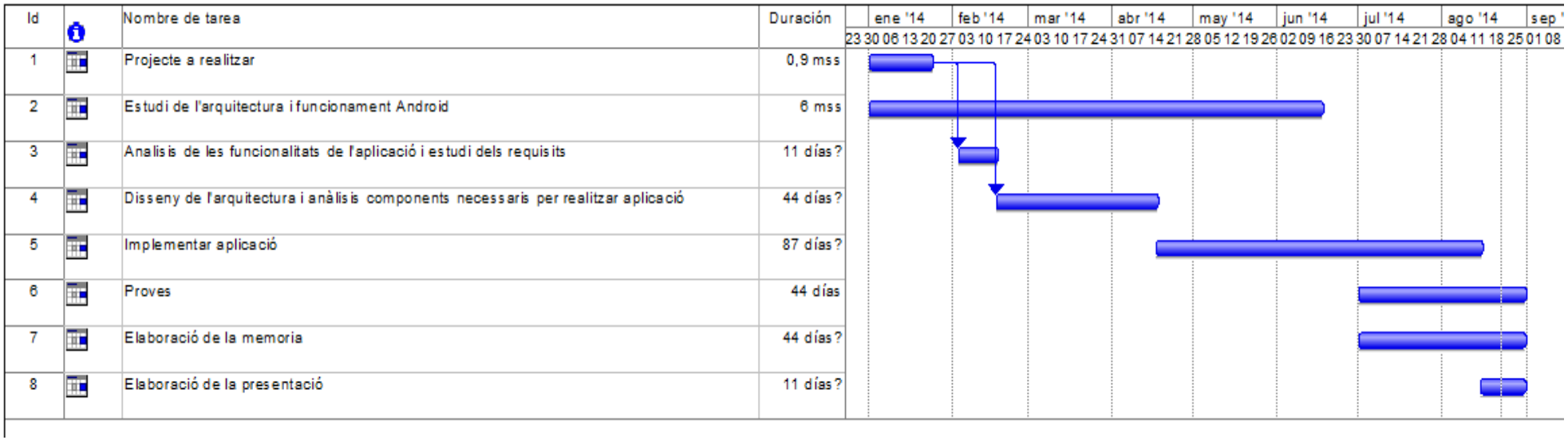

Figura 1: Planificació

Fent balanç s'han produït alguns retards més importants en certes fases del projecte i aquests són els següent:

- Decidir quin servidor utilitzar per fer de servidor de base de dades entre *Google*, *Amazon* i *Apache* amb *PHP* i *MySQL*.

- Problemes amb la localització dels errors del programa, en detectar en quina part del codi estaven (part client o servidor) i trobar les eines per facilitar aquesta feina

- Tractament d'imatges.

- Compaginar la feina a temps complert amb la realització del projecte.

1.5. Estructura del document

Aquest document està format per diferents capítols que detallen les fases de desenvolupament del projecte. A continuació es resumiran breument per tindre una idea de què conté cada part.

- Capítol 1. Introducció i Objectius: conté un petit resum del projecte indicant quina motivació a portat a la realització del projecte i els objectius del mateix.

- Capítol 2. Tecnologies utilitzades: descriu les tecnologies utilitzades per la realització del projecte.

- Capítol 3. Anàlisi de requeriments: descriu dels requeriments de l'aplicació funcionals i no funcionals, mostra i detalla els casos d'ús i els diagrames de classes.

- Capítol 4. Disseny de l'aplicació: descriu el model relacional utilitzat en la base de dades del servidor web. També descriu com s'ha dissenyat la interfície amb la que interactua l'usuari, els estils de diàleg, el disseny de les pantalles i menús, les icones i metàfores i la pantalla inicial.

- Capítol 5. Implementació: descriu les eines utilitzades i les decisions preses durant aquesta fase. I com s'ha realitzat el desenvolupament de l'aplicació.

- Capítol 6. Conclusions i treball futur: descriu les conclusions del projecte i les possibilitats de millora o noves funcionalitats que es poden afegir a l'aplicació.

- Annex 1: Manual d'Usuari: descriu el manual d'usuari

- Annex 2: Instal·lació *ADT*: descriu la instal·lació del *ADT* de *Android* en *Eclipse*.

- Annex 3: *API Google Maps*: descriu com obtenir una clau, instal·lar i utilitzar el *API de Google*.

- Annex 4: Diagrama de classes detallat: Mostra el diagrama de classes amb més detall que el mostrat en l'apartat de anàlisi de requeriments.

- Annex 5: Especificació d'alt nivell de la resta de casos d'ús: mostra la resta de casos d'ús que no s'han especificat en l'apartat d'anàlisi de requeriments.

Capítol 2: Tecnologies Utilitzades

2.1. Descripció de les tecnologies utilitzades.

Les tecnologies utilitzades en el projecte han estats les següents:

- Servidor Web *Apache*
- Llenguatge *PHP*
- Base dades *MySQL*
- *Android* amb base de dades *SQLite*

2.1.1. *PHP*

PHP és un acrònim que significa *PHP Hypertext Pre-processor* (inicialment *PHP Tools* o *Personal Home page Tools*). Va crear-lo originalment Rasmus Lerdorf, però la implementació principal de *PHP* es produïda ara per *The PHP Group* i serveix com l'estàndard de fet per *PHP* al no haver-hi una especificació formal. Publicat sota la *PHP License, la Free Software Foundation* considera aquesta llicència com software lliure [1].

Les característiques principals son:

- Orientat al desenvolupament d'aplicacions web dinàmiques amb accés a informació emmagatzemada en una base de dades.
- És considerat un llenguatge fàcil d'aprendre, ja que en el seu desenvolupament es van simplificar diferents especificacions, com és el cas de la definició de les variables primitives.
- El codi font escrites *PHP* és invisible al navegador web i al client, ja que és el servidor el que s'encarrega d'executar el codi i enviar el resultat *HTML* al navegador. Això fa que la programació en *PHP* sigui segura i confiable.
- Capacitat de connexió amb la majoria dels motors de bases de dades que s'utilitzen en l'actualitat, destaca la seva connectivitat amb *MySQL* y *PostgreSQL*.
- Capacitat d'expandir el seu potencial utilitzant mòduls (anomenats ext's o extensions).

- Té una documentació molt amplia en el seu lloc web oficial, entre la qual destaca el fet que totes les funcions del sistema estan explicades i exemplificades en un únic arxiu d'ajuda.

- Es lliure, per tant una alternativa de fàcil accés a tothom.

- Permet aplicar tècniques de programació orientada a objectes.

- No requereix definició de tipus de variables, encara que totes les seves variables es poden avaluar també pel tipus que estan manejant en temps d'execució.

- Disposa de control d'excepcions (des de *PHP5*).

2.1.2. Servidor *Apache*

El servidor *HTTP Apache* és un servidor web *HTTP* de codi obert, per plataformes *UNIX*, *Microsoft Windows*, *Macintosh* i altres, que implementa el protocol *HTTP/1.1* i la noció de lloc virtual. [2]

El servidor *Apache* es desenvolupa dins del projecte *HTTP Server* (*httpd*) de la *Apache Software Foundation*.

Apache presenta entre altres característiques altament configurables, bases de dades d'autentificació i negociat de contingut, però va ser criticat per la falta d'una interfície gràfica que ajudi a configurar-lo.

Els principals avantatges són:

- Modular

- Codi Obert

- Multi-plataforma

- Extensible

- Popular

Apache és utilitzat principalment per enviar pàgines web estàtiques i dinàmiques en el *World Wide Web*. Moltes aplicacions web estan dissenyades assumint que s'utilitzarà el *Apache* com a servidor web, o moltes característiques pròpies d'aquest servidor.

Apache s'utilitza per moltes altres tasques on el contingut necessita posar-se a disposició de forma segura i confiable.

La llicència de software sota la qual el software de la fundació Apache es distribuït és una part distintiva de la història de *Apache HTTP Server* i de la comunitat de codi obert. La llicència *Apache* permet la distribució de derivats de codi obert i tancat a partir del codi font original.

2.1.3. MySQL

MySQL és un sistema de gestió de bases de dades relacional, multi fil i multi usuari amb més de sis milions d'instal·lacions. Des de gener de 2008 és una subsidiària de Sun Microsystems, que alhora ho és de *Oracle Corporation* des d'abril de 2009 i amb el nom de *MySQL AB* desenvolupa software lliure amb un esquema de llicenciament dual.

MySQL està patrocinat per una empresa privada, que te el copyright de la major part del codi.

Les característiques principals són:

Inicialment, *MySQL* no tenia els elements considerats essencials en les bases de dades relacionals, com transaccions i integritat referencial, però tot i això va atraure als desenvolupadors de pagines web de contingut dinàmic per la seva simplicitat.

Però poc a poc els elements que no tenia s'estan incorporant, ja sigui per desenvolupadors interns, com per desenvolupadors de software lliure. Les característiques incorporades en les últimes versions són:

- Ampli subconjunt del llenguatge *SQL*
- Disponibilitat en gran quantitat de plataformes i sistemes.
- Possibilitat de selecció de mecanismes d'emmagatzematge que ofereixen diferents velocitats d'operació, suport físic, capacitat, distribució geogràfica, transaccions.
- Transaccions i claus foranies.
- Connectivitat segura.
- Replicació
- Cerca i indexació de camps de text.

Les característiques distintives de *MySQL* són:

- Permet escollir entre múltiples motors d'emmagatzemament per cada taula. Per exemple: els natius *MyISAM, InnoDB,* desenvolupats per partners com *solidDB, NitroEDB...* i desenvolupats per la comunitat com memcache, httpd...

- Agrupació de transaccions, reunint múltiples transaccions de varies connexions per augmentar el numero de transaccions per segon.

2.1.4. *Android*

Android és un sistema operatiu basat en el *kernel* de *Linux*, dissenyat principalment per dispositius mòbils amb pantalla tàctil, com telèfons intel·ligents o tabletes, i també per rellotges intel·ligents, televisors ... Inicialment va ser desenvolupat per *Android Inc*., que *Google* va ajudar econòmicament i més tard va comprar-la. Es va presentar al 2007 junt amb la fundació del *Open Handset Alliance* (un consorci de companyies de hardware, software i telecomunicacions) per avançar en els estàndards oberts dels dispositius mòbils.

Les característiques principals són:

- Plataforma realment oberta. és una plataforma de desenvolupament lliure basada en *Linux* i de codi obert.

- Adaptable a qualsevol tipus de hardware.

- Portabilitat assegurada. Les aplicacions finals són desenvolupades en *Java*, i això ens assegura que podran ser executades en qualsevol tipus de CPU, ja que s'aconsegueix gracies al concepte de màquina virtual.

- Arquitectura basada en components inspirats en Internet. Per exemple, el disseny de la interfície d'usuari es fa en *xml*, i això permet que la mateixa aplicació es pot executar en pantalles de diferents mida s.

- Filosofia de dispositiu sempre connectat a Internet.

- Gran quantitat de serveis incorporats. Per exemple, localització basada en GPS o xarxes, bases de dades amb SQL, reconeixement de veu, navegador, multimèdia.

- Nivell de seguretat acceptable. Els programes es troben aïllats uns dels altres gracies al concepte d'execució dins d'una capsa heretat de Linux. A més, cada aplicació disposa d'una sèrie de permisos que permeten el seu rang d'actuació.

- Optimitzat per baixa potència i poca memòria. *Android* utilitza la *Màquina Virtual Dalvik*. Es tracta d'una implementació de *Google* de la màquina virtual de Java optimitzada per dispositius mòbils.

- Alta qualitat de gràfics i so. Gràfics vectorials suavitzats, animacions inspirades en *Flash*, gràfics en 3 dimensions basats en *OpenGL*. Incorpora *codecs* estàndard més comuns de àudio i vídeo.

2.1.4.1. Arquitectura sistema Android.

El següent gràfic mostra l'arquitectura d'*Android*. Formada per quatre capes. Una de les característiques més importants és que totes les capes estan basades en software lliure.[4]

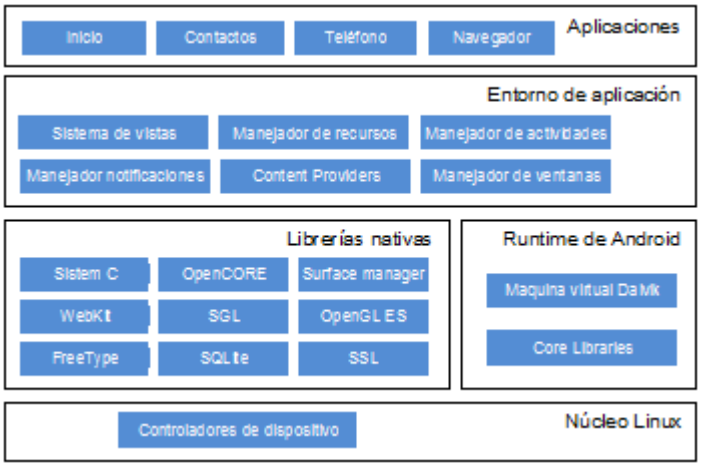

Figura 2: Arquitectura Android

2.1.4.2. Kernel de Linux.

El nucli d'*Android* està format pel sistema operatiu *Linux* versió 2.6. Proporciona serveis com la seguretat, el maneig de la memòria, el multiprocés, la pila de protocols i el suport de *drivers* per dispositius.

Aquesta capa del model actua com la capa d'abstracció entre el hardware i la resta de la pila. Per tant, és la única capa depenent del hardware.

2.1.4.3. Android Runtime

Està basat en el concepte de màquina virtual utilitzat en *Java*. Donat les limitacions dels dispositius on ha de córrer *Android* (poca memòria i processador limitat) no va ser possible utilitzar una màquina virtual *Java* estàndard. *Google* va crear la màquina virtual *Dalvik*, per respondre millor a aquestes limitacions.

2.1.4.4. Llibreries Natives

Inclou un conjunt de llibreries en C/C++ utilitzades en varis components Android. Estan compilats en codi natiu del processador. Moltes de les llibreries utilitzen projectes de codi obert. Algunes d'aquestes llibreries son: *System C library, Media Framework, Surface Manager, WebKit, SGL, Librerias 3D, FreeType, SQLite, SSL*.

2.1.4.5. Entorn Aplicació

Proporciona una plataforma de desenvolupament lliure per aplicacions amb gran riquesa e innovacions (sensors, localització, serveis, barra de notificacions).

Aquesta capa ha estat dissenyada per simplificar la reutilització de components. Les aplicacions poden publicar les seves capacitats i altres poden fer ús d'elles (subjectes a les restriccions de seguretat). Aquest mateix mecanisme permet als usuaris substituir components.

Una de les més grans fortaleses de l'entorn d'aplicació d' *Android* és que s'aprofita el llenguatge de programació Java. El *SDK* d'*Android* no acaba d'oferir tot el disponible pel seu estàndard del entorn d'execució *Java (JRE*) però és compatible amb una fracció molt significativa de la mateixa.

Els serveis més importants son:

- ***Views***: extens conjunt de vistes (part visual dels components).
- ***Resource Manager*** proporciona accés a recursos que no són en codi.
- ***Activity Manager*** maneja el cicle de vida de les aplicacions i proporciona un sistema de navegació entre elles.
- ***Notification Manager***, permet a les aplicacions mostrar alertes personalitzades en la barra d'estat.
- ***Content Providers***, mecanisme senzill per accedir a dades de altres aplicacions (com els contactes).

2.1.4.6. Aplicacions

Aquest nivell està format pel conjunt d'aplicacions instal·lades en una màquina *Android*. Totes les aplicacions han de córrer en la màquina virtual *Dalvik* per garantir la seguretat del sistema.[5]

Components bàsics d'una aplicació *Android*.

Figura 3: Components aplicació Android.

2.1.4.6.1. Android Manifest

El *Android Manifest* és un fitxer encarregat de comunicar-li al sistema operatiu *Android*:

- Els components què disposa l'aplicació.
- Els permisos necessaris per l'aplicació (si utilitzarà la camara, el GPS…)
- La versió d'*Android* mínima necessaria perquè l'aplicació funcioni.
- Els hardware i software requerit i/o utilitzat
- Les llibreries externes què utilitza

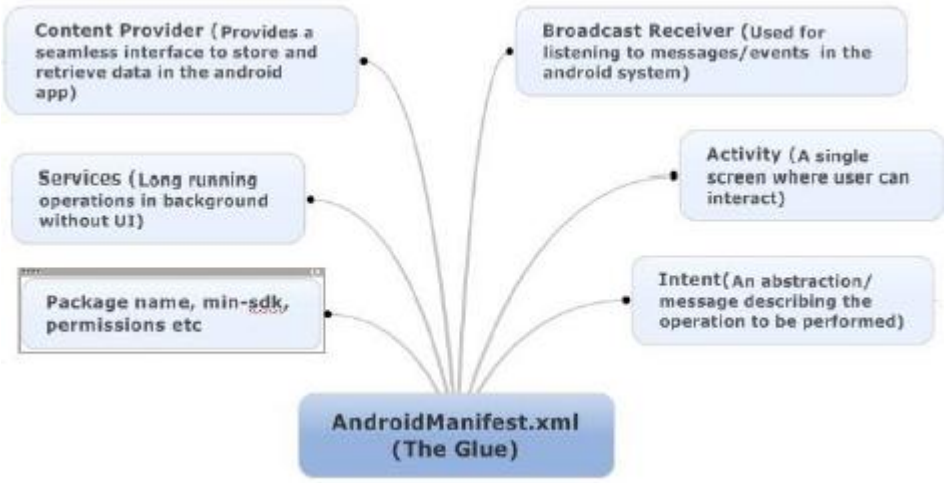

Figura 4: Android Manifest

2.1.4.6.2. Activity

Una aplicació en *Android* està formada per un conjunt d'elements bàsics de visualització, les pantalles de l'aplicació. En *Android* cada un d'aquests elements o pantalles es coneixen com a activitats La seva funció principal és la creació del interfície d'usuari. Les diferents activitats creades seran independents entre sí, encara que totes treballaran per un objectiu comú. Tota activitat ha de pertànyer a una classe descendent de *Activity*. El cicle de vida d'un *activity* és el següent:

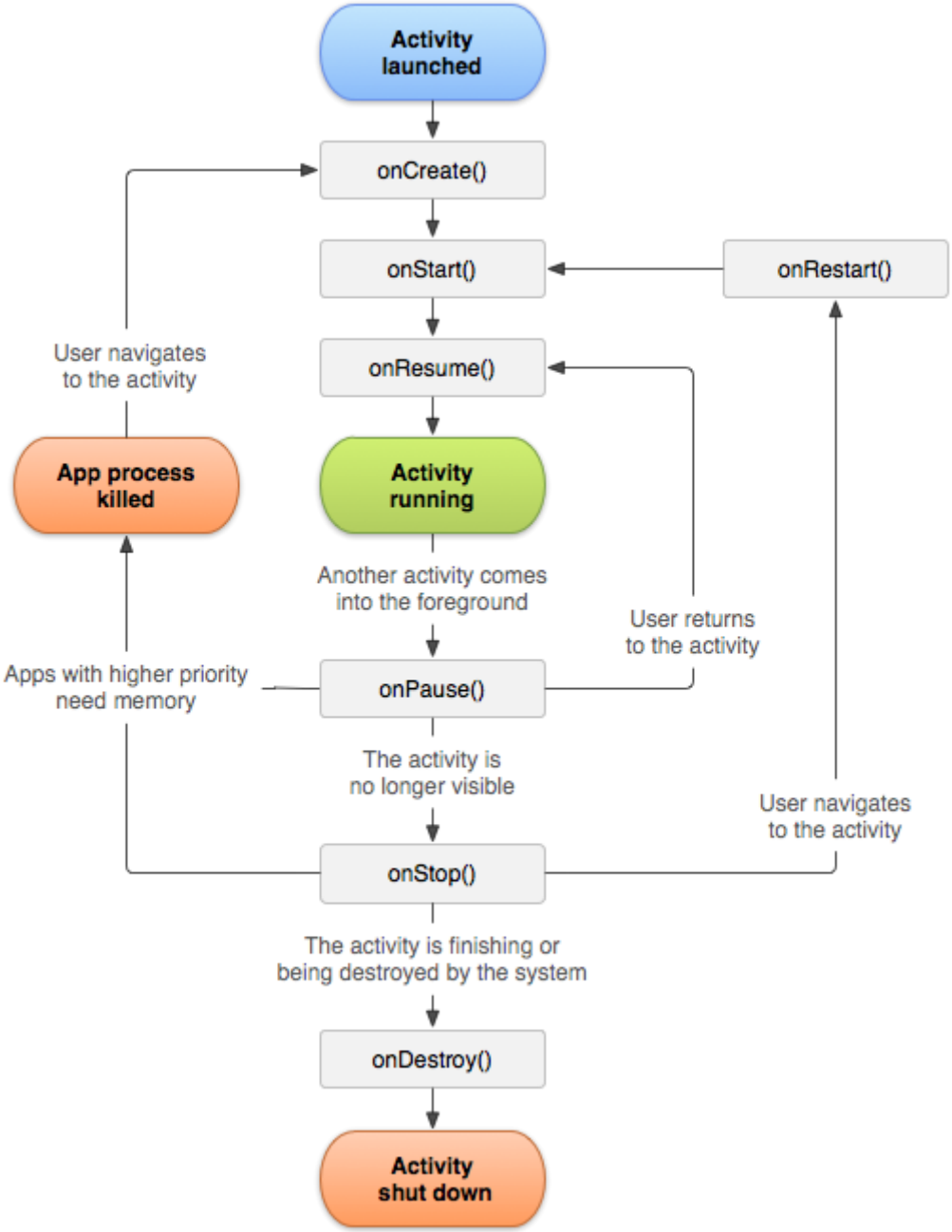

Figura 5: Cicle de vida d' una Activity

Una aplicació està composada per varies activitats, vinculades entres elles d'alguna manera. Com desenvolupadors es pot llençar una activitat a partir d'un altra, quedant-se la primera en pausa. El sistema manté en memòria una cua, que s'anomena back stack, que consisteix en una cua *LIFO*, perquè quan es polsi el botó enrere, el sistema trau l'activitat actual i mostrarà la anterior de la cua. Encara que aquest comportament pot ser modificat segons el nostre interès.

2.1.4.6.3. *Services*

Els serveis son components sense interfície gràfica que s'executen en segon pla. Poden realitzar qualsevol tipus d'accions, com actualitzar dades, llançar notificacions, o llançar activitats si es necessita interaccionar amb l'usuari.

2.1.4.6.4. *Content Provider*

Un proveïdor de continguts (content provider) és el mecanisme que s'ha definit en *Android* per compartir dades entre aplicacions. Mitjançant aquests components és possible compartir determinats dades de la nostra aplicació sense mostrar detalls sobre com te emmagatzemades les dades, com està estructurada l'aplicació o com és la seva implementació. També podrem accedir a dades d'un altra aplicació a través dels *content provider* que hagi definit aquesta.

2.1.4.6.5. *Widget*

Els *widgets* són elements visuals, normalment interactius, que es poden mostrar en la pantalla principal del dispositiu *Android* y rebre actualitzacions periòdiques. Permeten mostrar informació de l'aplicació al usuari directament sobre la pantalla principal.

2.1.4.6.6. Intent

Un *intent és* l'element bàsic de comunicació entre els diferents components *Android* que s'han descrit anteriorment. Es poden entendre com els missatges o peticions que s'envien entre els diferents components d'una aplicació o entre diferents aplicacions. Mitjançant un *intent* es pot mostrar una activitat des de qualsevol altra, iniciar un servei.

Hi ha dos tipus de *intent*: explícit (quan coneixem exactament el component que hem de llançar i l'especifiquem a través del nom del component) i implícit (on no coneixem el nom del component però sabem l'acció que volem executar i li demanem al sistema operatiu que busqui quina aplicació pot executar la nostra acció i en cas de haver-n'hi varies ens donarà a elegir amb quina es vol fer l'acció).

En el projecte els intent explícits s'han utilitzar per llançar les pantalles específiques de l'aplicació, mentre que els implícits són aquells que s'han utilitzat per llançar les funcions disponibles en el mòbil com fer la trucada de telèfon al seleccionar el telèfon de la botiga, o enviar un correu electrònic al seleccionar l'adreça de correu electrònic...

Capítol 3. Anàlisi de requeriments.

Els requeriments especifiquen què ha de fer el sistema (les funcions) i les seves propietats essencials i desitjables. La captura dels requeriments te com a objectiu principal la comprensió de què és el que volen els clients i que esperen els usuaris que faci el sistema. Un requeriment expressa el propòsit del sistema sense considerar com s'ha implementar.

L'anàlisi de requeriments és el conjunt de tècniques i procediments que ens permeten conèixer els elements necessaris per definir un projecte de software. Permet especificar les característiques operacionals del software, indicar la interfície del software amb altres elements del sistema i establir les restriccions que ha de complir el software.

Els requeriments es classifiquen en requeriments funcionals i no funcionals.

- Els requeriments funcionals d'un sistema descriuen el que ha de fer el sistema.

- Els requeriments no funcionals descriuen les restriccions sobre el sistema que limita la forma de resoldre un problema. Restringeixen els serveis o funcions ofertes pel sistema, inclouen restriccions de temps, tipus de procés de desenvolupament a utilitzar, fiabilitat, temps de resposta, capacitat d'emmagatzemament. Aquests requeriments posen límits i restriccions al sistema.

En aquest capítol es descriuran els requeriments definits per poder crear l'aplicació final. Es mostrarà el diagrama de casos d'ús i la descripció d'alt nivell d'aquests i el diagrama de classes de l'aplicació.

3.1. Requeriments funcionals.

Usuari no registrat

- Donar-se d'alta a l'aplicació.

- Obtenir informació dels plats de la botiga.

- Consultar el plat del dia/recomanació del xef.

- Obtenir informació de la botiga (telèfon, adreça, mail, web).

- Obtenir la localització de la botiga i del usuari i, si es possible, una ruta fins la botiga.

- Consultar l'ajuda de l'aplicació.

- Consultar la versió i copyright de l'aplicació.

- Consultar la configuració de l'aplicació i/o canviar-la.
- Crear una comanda i/o modificar-la
- Encriptar dades password abans d'enviar la informació al servidor.

Usuari registrat:

- Identificar-se a l'aplicació
- Rebre notificació quan esta llesta la comanda
- Activar/Desactivar plats visibles pel client.

Usuari Administrador:

- Fer una recomanació del plat del dia.
- Crear nous plats.
- Consultar les comandes dels clients.
- Enviar notificació conforme la comanda ja està preparada.

3.2. Requeriments no funcionals

Llistat de requeriments no funcionals, categoritzats segons la classificació tradicional

3.2.1. Producte

3.2.1.1. Usabilitat

- Crear una interfície amigable amb l'usuari, intuïtiva i fàcil d'utilitzar.

3.2.1.2. Portabilitat

- L'aplicació és compatible amb mòbils i tablets que tenen des de la versió 2.2.3 fins a la versió 4.4.
- L'aplicació és compatible amb pantalles de mòbils 4.3'' fins a pantalles de tablets 10.1'' (i de les mides intermèdies).

3.2.2. Eficiència

3.2.2.1. Rendiment

- L'aplicació necessita una bona connexió a Internet per aconseguir millorar el rendiment de les consultes al servidor. El rendiment varia segons la qualitat de la connexió a Internet.

3.2.2.2. Espai

- L’aplicació necessita un tenir disponible en el sistema una targeta SD externa on s’emmagatzemen les fotografies dels plats què apareixen en l’aplicació.

3.2.3. Externs

3.2.3.1. Legislatiu

3.2.3.1.1. Privacitat

- Protegir dades de l’usuari amb la comunicació amb el servidor.

3.2.3.1.2. Seguretat

- Control d’accés a les dades de l’aplicació, les dades individuals només les ha de poder veure l’usuari i l’administrador, no han d’estar accessibles a la resta d’usuaris que utilitzi l’aplicació.

3.3. Casos d’ús.

3.3.1. Actors

Els actors de la nostra aplicació són els següents:

- ***Usuari***: són totes aquelles persones que poden accedir a l’aplicació i poden consultar les dades bàsiques, com quins plats s’ofereixen, on està la botiga, com es pot contactar amb ella, com es pot arribar.

- ***Usuari registrat***: són totes aquelles persones que poden accedir a l’aplicació igual que un usuari, però que poden realitzar comandes i rebre la notificació quan la comanda està llesta.

- ***Administrador***: són les persones que tenen accés a l’aplicació i poden fer el mateix que la resta de usuaris i usuaris registrats però tenen una sèrie de funcions pròpies com afegir nous plats, treure plats no disponibles, indicar plat del dia, i enviar missatge conforme comanda esta preparada.

- ***Servidor web:*** servidor web on es guarden les dades i es realitzen les consultes necessàries pel funcionament de l’aplicació.

3.3.2. Casos d' ús

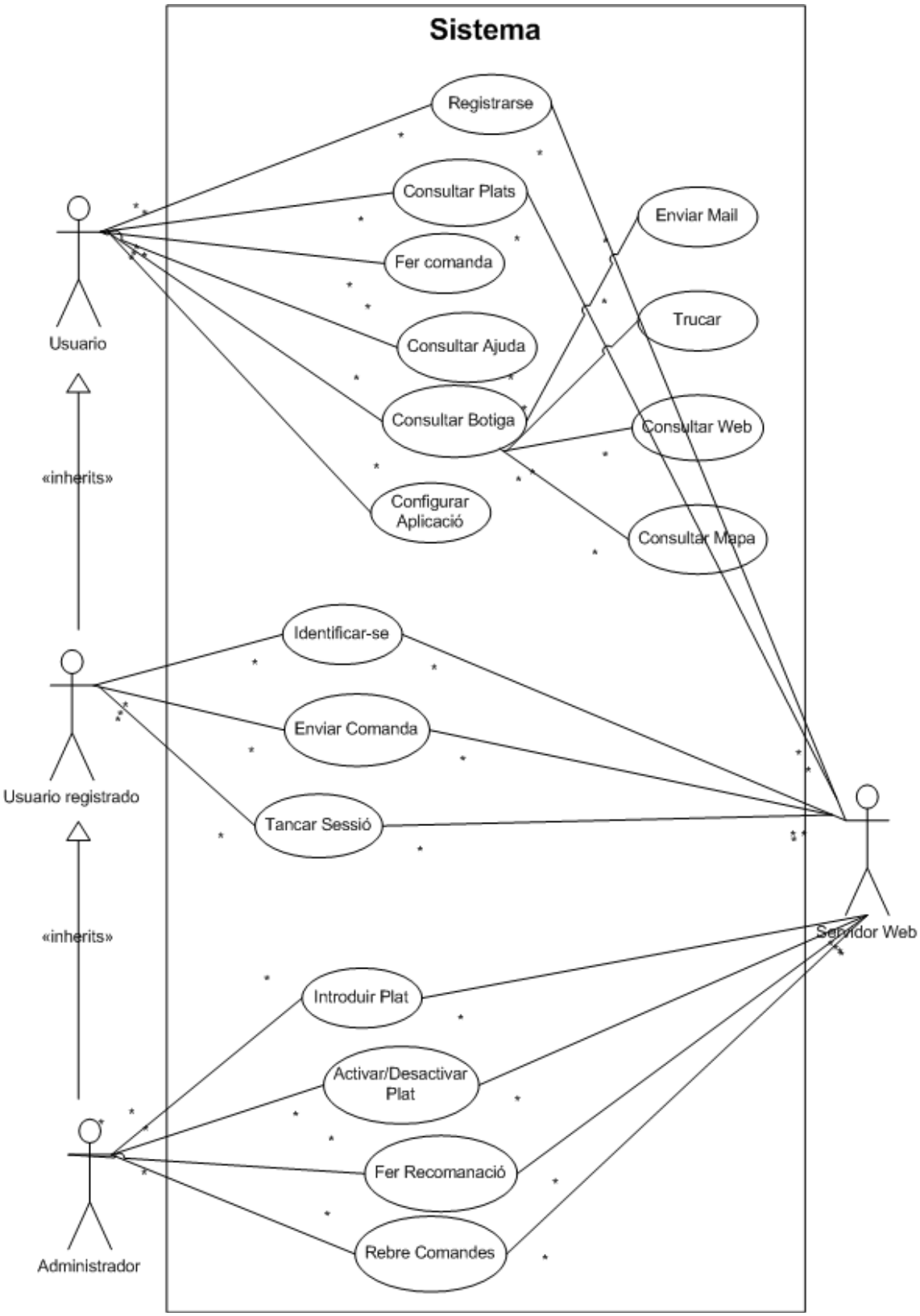

Figura 6: Diagrama de casos d'ús.

3.3.3. Especificació d'alt nivell

A continuació es detallaran alguns dels casos d'ús de l'usuari, la resta es podran consultar en l'Annex 5 del projecte:

Descripció de casos d'ús de l'usuari

Taula 1: Registrar-se

Nom	Registrar-se	
Descripció	Realitza un registre en el mòbil i en la Base de Dades del servidor per identificar a l'usuari de forma única	
Actors	Usuari i servidor web	
Pre-condicions	No ha d'existir un usuari autentificat en el mòbil	
Flux normal	***Accions dels Actors*** 1.L'usuari no registrat omple els camps de usuari, nom i contrasenya i repetir contrasenya 2. L'usuari no registrat selecciona el botó Registro	***Accions del sistema*** 3. El sistema comprova que la contrasenya i la repetició són iguals, si no llença avís i no deixa continuar fins que no són iguals 4.el sistema envia les dades del usuari al servidor web 5. Comprovar que la identificació és única (Servidor Web) si la identificació és única el sistema grava al usuari en la base de dades. En cas contrari, dona error 6. El sistema envia el resultat de les comprovacions del servidor al mòbil i envia un missatge al usuari indicant el resultat de l'operació
Excepcions	Existeix el usuari que s'està intentant donar d'alta. El sistema indica al usuari en un missatge que el usuari que està posant ja esta donat d'alta en l'aplicació.	
Postcondicions	Creem l'usuari a la Base de dades	

Taula 2: Consultar Plats

Nom	Consultar Plats	
Descripció	Consulta de plats disponibles a la botiga	
Actors	Usuari, servidor web	
Pre-condicions	Cap	
Flux normal	***Accions dels Actors***	***Accions del sistema***
	1.L'usuari no registrat entra per defecte a la pantalla on pot consultar tots els plats quan entra en l'aplicació. 2.L'usuari no registrat pot anarr al menú Plat per accedir a la pantalla de la llista de plats disponibles	3.En qualsevol de les 2 accions de usuari el sistema realitza una consulta a la base de dades interna del mòbil (que ha estat omplerta amb les dades del servidor quan s'ha iniciat l'aplicació) per obtenir la llista de plats.
	4. L'usuari no registrat al seleccionar un element de la llista de plats es mostra una pantalla amb tota la informació disponible sobre aquell plat.	
Excepcions	Si no esta correctament carregada la llista, o les fotos dels plats encara s'estan baixant, apareixeran únicament les dades disponibles en aquell moment.	
Postcondicions	Consultar Plat / Fer comanda	

Taula 3: Fer Comanda

Nom	Fer comanda	
Descripció	Preparar comanda	
Actors	Usuari	
Pre-condicions	Estar en la pantalla de la llista de plats disponibles	
Flux normal	***Accions dels Actors***	***Accions del sistema***
	1.L'usuari no registrat pot afegir plats a una comanda des de la pantalla de llista mitjançant el botó corresponent	
		2.El sistema indica que s'ha afegit el plat x a la comanda i la quantitat que hi ha en aquell moment. Grava el registre del plat en la base de dades del mòbil per poder consultar-lo.
	3.L'usuari no registrat pot afegir plats i traure'ls de la comanda mitjançant el botó corresponent.	
		4.El sistema comprova que hi ha plats del seleccionat a la comanda i si és així treu un de la comanda si no indica que no hi ha plats disponibles i grava la modificació en la base de dades interna.
Excepcions	Si al intentar traure un plat d'una comanda no existeix aquell plat es produeix un error indicant que no existeix el plat	
Postcondicions	Si no existeix cap detall amb aquest plat el crea amb una quantitat de 1 i si ja existeix li suma 1 a la quantitat guardada en la BBDD interna	

Descripció de casos d'ús de l'usuari registrat.

Taula 4: Identificar-se

Nom	Identificar-se	
Descripció	Pregunta a l'usuari el usuari i contrasenya per deixar accedir a l'aplicació	
Actors	Usuari, servidor web	
Pre-condicions	Ha d'existir el usuari registrat prèviament	
Flux normal	***Accions dels Actors***	***Accions del sistema***
	1.L'usuari registrat ha d'omplir les dades demanades de usuari i contrasenya 2.L'usuari selecciona el botó Entrar	3.El sistema envia les dades introduïdes al servidor web. 4. El servidor web comprova que l'usuari i la contrasenya són correctes. 5. El sistema retorna una resposta indicant que el resultat de la comprovació.
Excepcions	Usuari i/o contrasenya no coincideixen amb els que hi ha donats d'alta	
Postcondicions	Si l'usuari es correcte es va a la llista de plats.	

Taula 5: Enviar Comanda

Nom	Enviar comanda	
Descripció	Acabar de revisar la comanda ja preparada i enviar la comanda a la botiga	
Actors	Usuari registrat i servidor web	
Pre-condicions	Estar Identificat	
Flux normal	***Accions dels Actors***	***Accions del sistema***
	1.L'usuari entra en la pantalla de la comanda.	
		2. el sistema comprova que l'usuari estigui correctament identificat, si no ho està mostra la pantalla de identificació perquè ho faci
	3. L'usuari pot veure el nom, mail i data de qui fa la comanda i els plats que s'han demanat 4.L'usuari pot veure la fitxa del plat seleccionant sobre l'element de la llista de plats de la pantalla comanda 5.L'usuari pot modificar els plats i les quantitats de la comanda 6.L'usuari selecciona sobre el botó enviar per enviar la comanda	
		7.El sistema envia la comanda i els plats seleccionats al servidor 8.El servidor web rep les dades i les emmagatzema a la base de dades
Excepcions	Si no es grava correctament les dades de la comanda al servidor dona error.	
Postcondicions	Enviar la comanda a la Base de Dades del servidor i gravar-la.	

Descripció de casos d'ús de l'usuari administrador

Taula 6: Crear plat nou

Nom	Crear nou plat	
Descripció	Afegir un plat al menú	
Actors	Usuari registrat com administrador i servidor web	
Pre-condicions	Estar Identificat com administrador	
Flux normal	***Accions dels Actors***	***Accions del sistema***
	1.L'usuari administrador selecciona el menú nou plat	
		2. El sistema obre la pantalla on permet donar d'alta un nou plat
	3. L'usuari administrador omple les dades demanades. També permet fer una foto o agafar una que ja esta guardada en el mòbil.	
		4.Al entrar una nova foto el sistema et permet seleccionar entre les opcions disponibles en el teu mòbil de fer una foto i obtenir-la de la galeria
	5. L'usuari administrador confirma	
		6. al confirmar el sistema envia les dades que s'han obtingut al servidor. 7.El servidor web emmagatzema les dades 8.El sistema envia resposta conforme les dades s'han emmagatzemat o no correctament en el servidor i el sistema ho mostra al usuari a través d'un missatge.
Excepcions	Les dades no s'han emmagatzemat correctament al servidor	
Postcondicions	Crear el plat en el servidor perquè els usuaris el puguin veure.	

Taula 7: Activar/Desactivar Plat

Nom	Activar/Desactivar plat	
Descripció	Activar/Desactivar un plat del menú	
Actors	Usuari registrat com administrador i servidor web	
Pre-condicions	Estar identificat com administrador	
Flux normal	***Accions dels Actors***	***Accions del sistema***
	1.L'usuari administrador entra en la llista de plats de l'administrador 2. L'usuari administrador marca o desmarca el check de la pantalla on indica activar/desactivar plat	3.El sistema envia la informació al servidor. 4.El servidor web guarda la informació a la base de dades. 5.El sistema envia informació si s'ha realitzat correctament el procés o no. Quan ho rep el mòbil indica que s'ha processat correctament o no al usuari.
Excepcions	La informació no s'ha actualitzat correctament al servidor	
Postcondicions	Activar/Desactivar el plat en el servidor perquè els usuaris no el puguin veure.	

Taula 8: Fer recomanació

Nom	Fer Recomanació	
Descripció	Recomanar un plat del menú	
Actors	Usuari registrat com administrador	
Pre-condicions	Estar Identificat com administrador	
Flux normal	***Accions dels Actors***	***Accions del sistema***
	1.L'usuari administrador entra en la llista de plats de l'administrador 2. L'usuari administrador marca o desmarca el check de la pantalla on indica recomanació	3.El sistema envia la informació al servidor. 4.El servidor web primer borra la recomanació anterior i guarda la nova recomanació a la base de dades. 5.El sistema envia informació si s'ha realitzat correctament el procés o no. Quan ho rep el mòbil indica que s'ha processat correctament o no al usuari
Excepcions	La informació no s'ha actualitzat correctament al servidor	
Postcondicions	Al enviar una recomanació el servidor borra la recomanació anterior i es queda amb la recomanació nova. Només pot haver una recomanació.	

3.4. Diagrama de classes

El diagrama de classes[20] de l'aplicació és el següent:

- Paquets que formen l'aplicació:

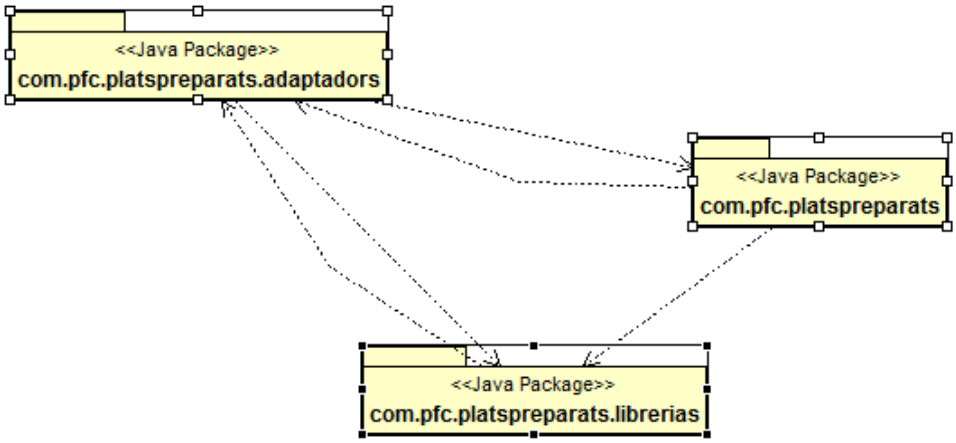

Figura 7: Diagrama de classes dels paquets que formen l'aplicació

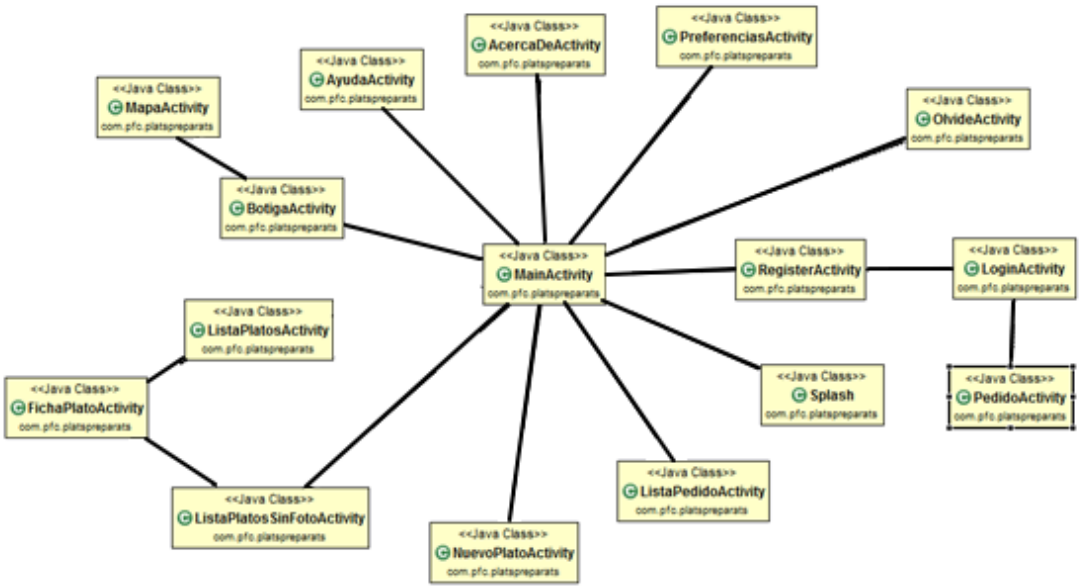

Figura 8: Diagrama de classes

Capítol 4: Disseny de l'aplicació

Després d'haver realitzat l'anàlisi de requisits i establir les funcionalitats de l'aplicació, s'ha de definir un disseny dins del cicle de desenvolupament del software. Es descriurà de forma detallada un disseny de l'arquitectura de l'aplicació, un disseny de la base de dades i es comentaran els diferents dissenys realitzats per la interfície amb l'usuari.

4.1. Disseny de l'arquitectura

En aquest apartat s'analitza l'arquitectura de l'aplicació i també un estudi dels components que formaran aquesta arquitectura. Això permetrà veure si es van complint els requisits i les funcionalitats demanades.

L'aplicació utilitza una arquitectura de disseny basada en Model-Vista-Controlador (MVC), de forma que les dades de l'usuari estan separades de la interfície que utilitza i de la lògica de control.

Aquest model és un patró d'arquitectura de software que separa les dades i la lògica de negoci d'una aplicació de la interfície d'usuari i el mòdul encarregat de gestionar els esdeveniments i les comunicacions. Per això MVC proposa la construcció de tres components diferents que són el model, la vista i el controlador, és a dir, per una part defineix components per la representació de la informació i per un altra per la interacció amb l'usuari. Aquest patró es basa en idees de reutilització de codi i separació de conceptes, característiques que busquen facilitar la tasca de desenvolupament d'aplicacions i el seu manteniment posterior. [18]

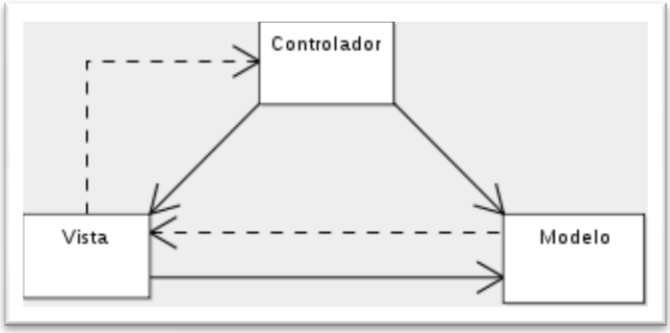

Figura 9: Model MVC

L'arquitectura elegida és un Servidor *Apache* amb una base de dades *MySQL* y servidor web realitzat amb *PHP*. A més d'això, quan l'aplicació vol obtenir el mapa de la localització s'ha de comunicar amb el servidor de *Google* per poder obtenir les dades.

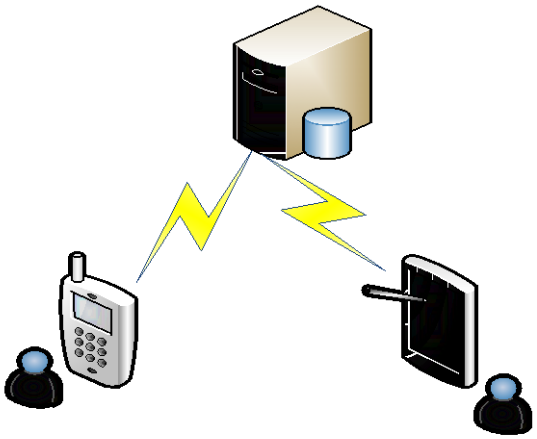

Figura 10: Arquitectura de l'aplicació

4.2. Disseny de la Base de Dades

La base de dades, tan pel servidor utilitzant *MySQL* com pel propi dispositiu mòbil amb *SQLite*, que hem creat és molt simple i es compon de quatre taules:

- Taula ***Login***: Taula on s'emmagatzemen les dades del client i de l'administrador de l'aplicació (indicat en el camp ***TipoUsuario***), l'usuari amb el que s'entra en l'aplicació (camp ***Usuario***) i la contrasenya que s'utilitza per comprovar que l'usuari és el registrat (camp ***Password***).

- Taula ***Plato***: Taula on hi ha emmagatzemats totes les dades dels plats disponibles en la botiga, amb un identificador del plat (camp ***IdPlato***), i totes les dades necessàries per descriure'l com són nom, descripció, ingredients, calories, foto, preu, recomanació del xef, i el nom del fitxer amb la foto del plat.

- Taula ***Pedido***: Taula on hi ha emmagatzemada la informació sobre la comanda, el identificador de la comanda (camp ***idPedido***) i la del client que ha fet la comanda (camp ***idCliente,*** que permet relacionar la comanda amb el client que l'ha fet), també s'indica la data de la comanda i la situació en que es troba la comanda (camp ***Situación*** que pot pendre com a valors 0 – pendent d'enviar, 1 – enviada, 2 – acceptada, 3 – preparada, 4 – recollida, 9 - anul·lada).

- Taula ***LineasPedido***: Taula on hi ha els plats i les quantitats d'aquests que s'han demanat en cada comanda, on s'identifica la comanda (camp i***dPedido*** que permet relacionar cada línea de comanda amb la comanda a la que corresponen), s'identifica el numero de línea de la comanda (camp ***idLinea***), el plat que s'encomana (camp ***idPlat*** que permet relacionar la línea de la comanda amb el plat seleccionat***)***, la quantitat (camp ***Cantidad***) i la situació (camp ***Situacion*** que pot agafar els valors 0 – pendent d'enviar, 1 – enviada, 2 – acceptada, 3 – preparada, 4 – recollida, 9 - anul·lada).

En totes les taules de la base de dades s'han creat els camps created_at i modified_at per saber quan s'han creat els registres i la data de la última modificació soferta.

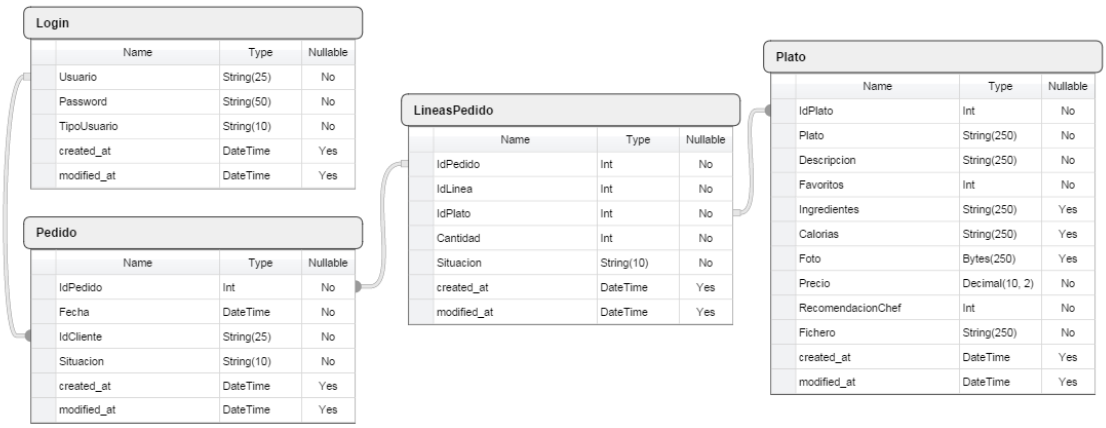

Figura 11: Diagrama de la base de dades

La eina que hem utilitzat per dibuixar aquest diagrama és *Sea Quail Database Diagram Tool*[17].

4.3. Disseny de la interfície

En el disseny de la interfície de l'usuari, s'han intentat aplicar una sèrie de principis bàsics que són els següents [10]:

- Familiaritat de l'usuari.
- Consistència.
- Mínima sorpresa.
- Recuperabilitat .
- Guia a l'usuari.
- Diversitat d'usuari.

4.3.1. Disseny de pantalles i menús.

S'ha intentat realitzar una presentació neta sense molts colors, en l'aplicació apareixen bàsicament el color blanc, gris i negre. Alguna icona te algun color més com la del carro de la compra que serveix per afegir un plat a la comanda que te un cercle de color verd amb un signe positiu, o el de traure un plat de la comanda, carro de la compra amb un cercle vermell amb un signe negatiu. El logo, les fotos dels plats i el mapa són els únics elements amb més colors de l'aplicació.

4.3.1.1. Menús

S'ha utilitzar la navegació a través de menús, per que l'usuari pugui navegar a través de l'aplicació.

Figura 12: Menús.

Un altra detall és que l'aplicació mostra 2 menús diferents segons l'usuari sigui administrador o un usuari normal- El menú per defecte és el del usuari normal i si al Identificar-se la resposta de la base de dades indica que és usuari administrador aleshores farà la crida al menú d'administrador.

En un usuari normal el menú que es mostra és el següent:

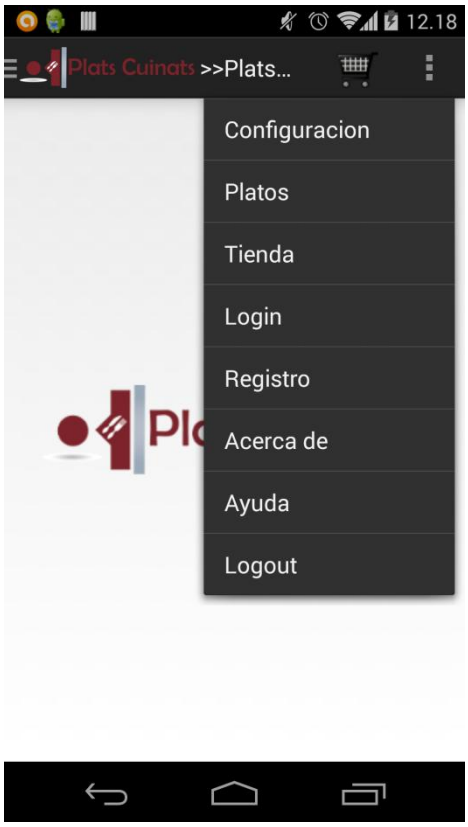

Figura 13: Menú usuari

I si és usuari administrador es:

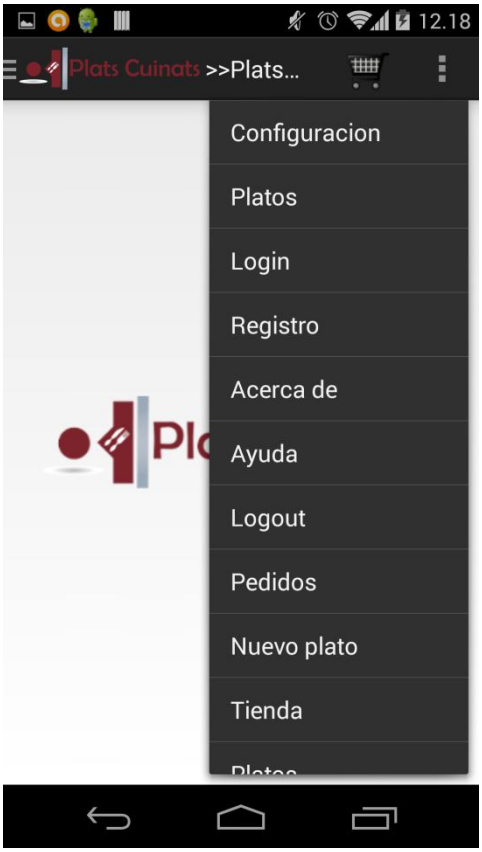

Figura 14: Menú Administrador

Això ho hem aconseguit utilitzant el codi següent sobreescrivint el procediment *onCreateOptionsMenu(Menu menu)*, que es el mètode que s'encarrega de crear els menús

```
public boolean onCreateOptionsMenu(Menu menu) {

    //if (!mNavigationDrawerFragment.isDrawerOpen()) {
        // Only show items in the action bar relevant to this screen
        // if the drawer is not showing. Otherwise, let the drawer
        // decide what to show in the action bar.
        /// buscar si usuari es administrador per enseñar un menu o un altre
    SQLiteHandler db = new SQLiteHandler(this.getApplicationContext());
    HashMap<String, String> usuario = db.getUserDetails();

    if (usuario.get("tipo").equals("ADM")) {

        supportInvalidateOptionsMenu();
        getMenuInflater().inflate(R.menu.amd, menu);
        //restoreActionBar();

    }
    else {
        supportInvalidateOptionsMenu();
        getMenuInflater().inflate(R.menu.usr, menu);
        restoreActionBar();
    }
    //}
    return super.onCreateOptionsMenu(menu);
}
```

Figura 15: Procedure OnCreateOptionsMenu(Menu menu)

I sobreescrivint el procediment *onPrepareOptionsMenu(Menu menu)* segons el tipus definit en l'usuari, que es el procediment que a partir de Android 3 s'encarrega de preparar els menús quan es volen canviar dinàmicament per codi:

```
@Override
public boolean onPrepareOptionsMenu(Menu menu) {

        SQLiteHandler db = new SQLiteHandler(this.getApplicationContext());
        HashMap<String, String> usuario = db.getUserDetails();

        if (usuario.get("tipo").equals("ADM")) {

            supportInvalidateOptionsMenu();
            getMenuInflater().inflate(R.menu.amd, menu);
            //restoreActionBar();

        }
        else {
            supportInvalidateOptionsMenu();
            getMenuInflater().inflate(R.menu.usr, menu);
            restoreActionBar();
        }
    return super.onCreateOptionsMenu(menu);
}
```

Figura 16: Procedure onPrepareOptionsMenu(Menu menu)

4.3.1.2. Pantalles i navegació

En algun cas les pantalles també tenen accessos a altres pantalles relacionades com és el cas de la pantalla de plats on al seleccionar sobre un plat anem a la fitxa del plat on es dona més informació sobre el plat.

Figura 17: Navegació entre la llista de plats i la fitxa

O la pantalla de identificació que permet anar directament a la pantalla de registre sinó estàs registrat o bé a la de "Olvidé mi contraseña" en el cas de no recordar la contrasenya.

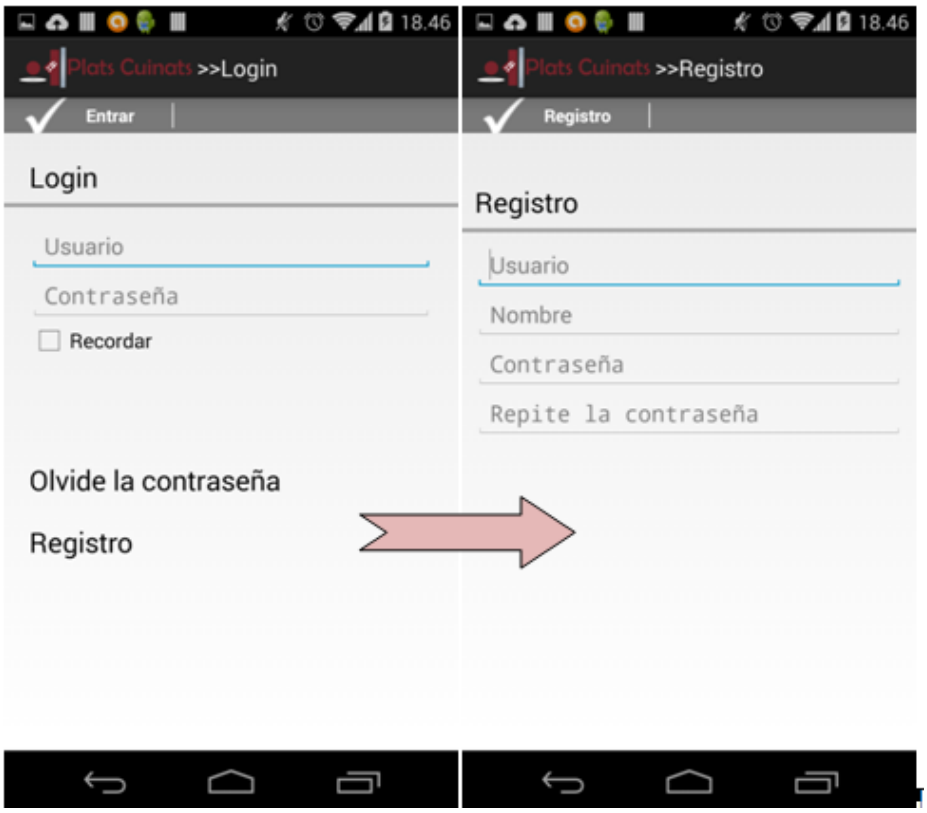

Figura 18: Navegació entre Login i Registre

O la de la botiga que et permet anar a la del mapa al seleccionar sobre la direcció.

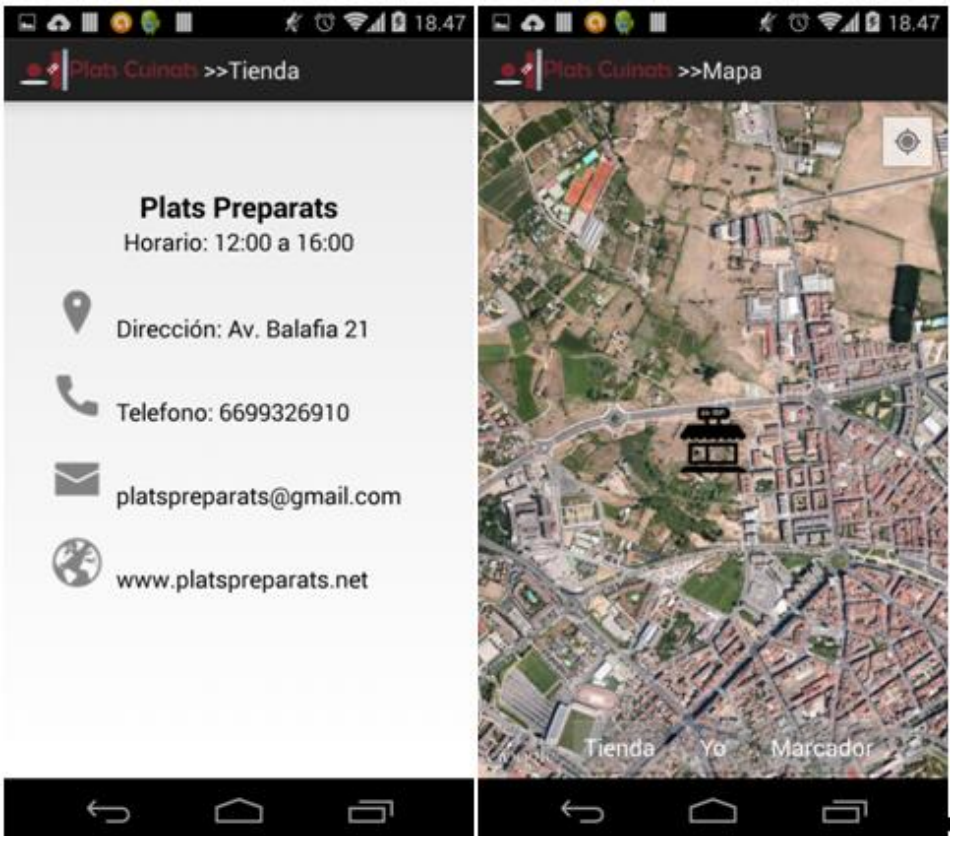

Figura 19: Navegació entre botiga i el mapa

4.3.2. Icones i metàfores.

S'han utilitzat icones que ja s'utilitzen en moltes altres aplicacions Android i que l'usuari ja està familiaritzat amb la seva funció. Per exemple:

-  : S'utilitza per anar a la pantalla de la comanda des de la pantalla principal.
-  : S'utilitza per afegir un element a la comanda tan en la pantalla on apareix la llista de plats com en la de la pròpia comanda.
-  : S'utilitza per traure un element (plat) de la comanda en els mateixos casos que l'anterior.
-  : S'utilitza per indicar que es desplega un menú
-  : S'utilitza per indicar que es pot trucar al telèfon indicat.
-  : S'utilitza per indicar que hi ha una web associada a l'adreça que l'acompanya.
-  : S'utilitza per indicar que es pot enviar un mail a l'adreça de mail que es mostra al seu costat.
-  : S'utilitza per indicar que es mostra una situació en un mapa.
-  : s'utilitza en el mapa per indicar la situació de la botiga.
-  : S'utilitza en el mapa per indicar la situació del "Jo" (persona que utilitza el telèfon.
-  : S'utilitza en els formularis per indicar el botó de confirmar que el formulari està omplert i confirmar les dades.
-  : S'utilitza per indicar que hi ha un menú lateral que es pot desplegar.

-  : s'utilitza per indicar la opció de configuració.
-  : s'utilitza per indicar informació.
-  : s'utilitza per indicar que es pot fer una foto o obtenir una imatge.

4.3.3. Estils de diàleg.

Els estils de diàleg que s'ha intentat aplicar son:

- Informar de totes les accions que fa l'usuari quan esta interaccionant amb l'aplicació a través d'un toast.
- Per exemple quan es fa la identificació, si no es troba l'usuari indica Login Error.
- Quan un usuari ja registrat intenta registrar-se i ja està registrat dona un error.
- Quan intenta enviar la comanda i no pot enviar-se correctament dona un error. I quan s'ha enviat correctament surt un quadre de diàleg felicitant-lo per haver realitzat la comanda correctament.
- En les accions que implica eliminar un registre s'ha programat un diàleg que permeti la confirmació d'aquesta acció. Això es troba en el cas de la comanda quan es vol eliminar un plat.
- Hi ha accions que permeten més d'una opció i es mostra a l'usuari una llista amb les diferents accions permeses per aquella opció com es el cas de si volen enviar un email, o trucar per telèfon.

En quan a la consistència s'ha intentat aplicar la mateixa manera de funcionar a totes les pantalles. En una llista si es realitza una selecció sobre un element de la llista s'obre la fitxa si està disponible.

En el cas de les pantalles on s'han de introduir i confirmar dades, per confirmar i enviar les dades el botó està situat sempre amb el mateix format i en el mateix lloc, sota la barra del títol, que és on hi ha el logo i ens indica en quina pantalla estem, encara que el text que defineix el botó no digui el mateix.

4.3.4. Pantalla inicial

La pantalla inicial s'ha dissenyat, seguint la resta de l'aplicació de forma senzilla, únicament s'ha utilitzat el logo de la botiga per fer la entrada a l'aplicació. Ja que s'ha intentat plasmar en el logo tot el que es fa en la botiga. El logo indica el nom de la botiga Plats Cuinats es pot veure un parell de rodones en representació dels plats i del moviment, i un parell de barres on en l'interior d'una d'elles hi ha una forquilla i un ganivet, en representació del menjar.

Aquesta pantalla a part de ser la imatge de la botiga s'utilitza per fer inicialment la càrrega de les dades des del servidor.

Figura 20: Pantalla inicial

Capítol 5: Implementació

5.1. Eines d'implementació

5.1.1. Mitjans emprats

Per realitzar el desenvolupament del projecte ha estat necessari l'ús dels següents elements de software:

- Sistema operatiu: *Windows 7 Professional 64 bits.*
- Servidor web: *Apache versió 2.2.14, MySQL versió 5.1.61, PHP versió 5.3*
- Eines de desenvolupament: *IDE Eclipse Kepler, Plugin ADT Android per Eclipse 4.3, PHPMyAdmin, extensió Chrome Postman Rest-client, extensió Chrome Sea Quail Database Diagram Tool,*
- Control de versions: *Eclipse Git Team Provider*
- Llibreries: *Java SDK 6, Android SDK rev. 4.3, API Google Maps v2 de Google*
- *Microsoft Office 2007 i Microsoft Visio2007 i Microsoft Project.*

5.1.1.1. Sea Quail Database Diagram Tool

Sea Quail Database Diagram Tool és una eina gràfica utilitzada per realitzar els diagrames de la base de dades. A partir d'aquests diagrames es poden generar els scripts de creació de les taules. [17]

5.1.1.2. PHPMyAdmin

PHPMyAdmin s'ha utilitzat per crear la base de dades en el servidor. és una eina escrita amb *PHP*, per manejar l'administració de *MySQL* a través de pàgines web. Es poden crear i eliminar bases de dades, crear, modificar i borra taules, crear, modificar i borra camps, executar sentencies *SQL*, administrar claus, administrar privilegis, exportar dades. Està disponible sota llicencia *GPL*.

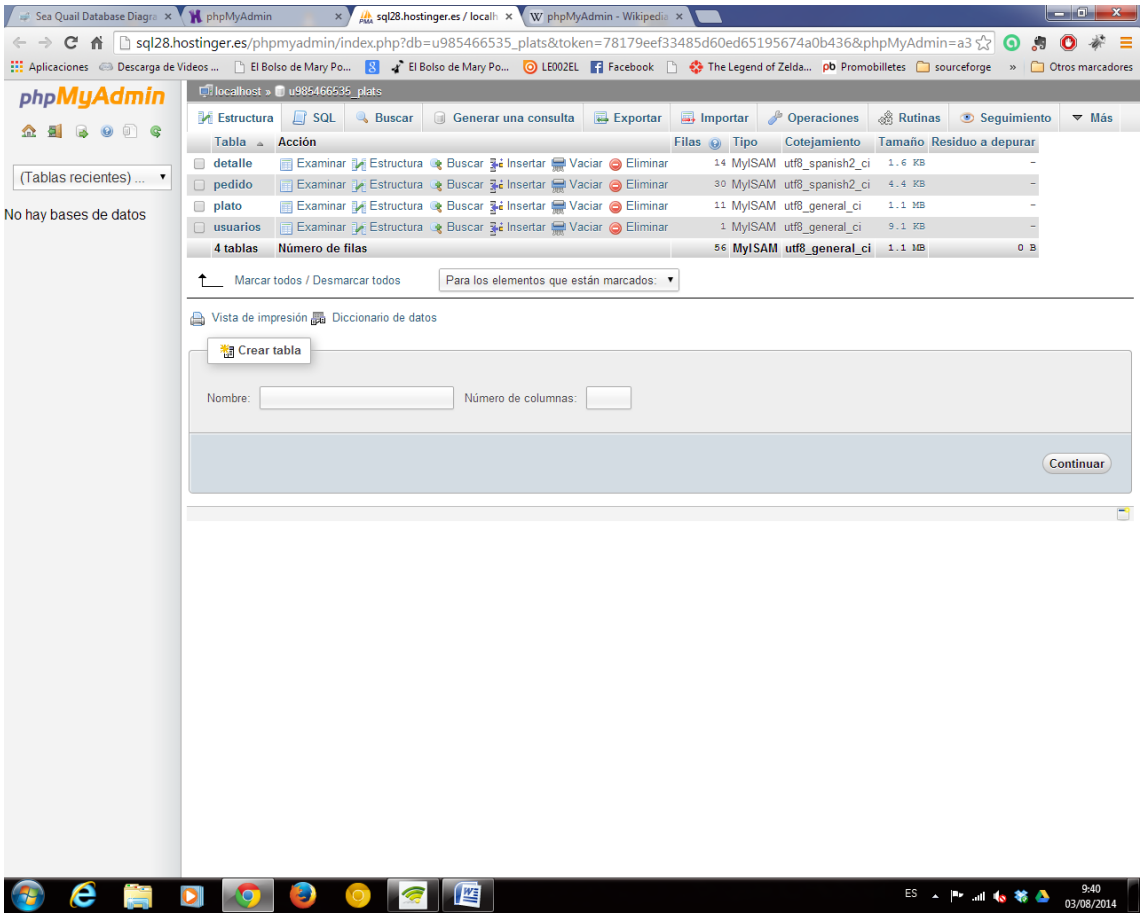

Figura 21: PHPMyAdmin

5.1.1.3. Postman REST-Client

És una extensió del navegador *Google Chrome*, que hem utilitzat per comprovar que tots els programes escrits amb *PHP* funcionin correctament.[12]

Característiques:

- Interfície compacta
- Peticions *HTTP* amb suport de carrega d'arxius
- Respostes amb format *JSON i XML*
- Respostes obertes com *HTML* en una nova finestra
- Suport *HATEOAS*
- Previsualització d'imatges
- Històric de peticions
- Ajudes bàsiques i OAuth 1.0

- Auto completat per *URL* i valors de capçalera
- Editors clau/valor per afegir paràmetres or valors de capçalera.
- Utilitza variables d'entorn per simplificar canvi entre configuracions
- Utilitza variables globals per valors comuns entre *APIs*.
- Utilitza la característica de vista prèvia per veure les variables i els seus valors abans de utilitzar-les.
- Dreceres de teclat per maximitzar la productivitat.

5.1.1.4. SDK Android

El *SDK* (*Software Development Kit*) d'*Android*, inclou un conjunt d'eines de desenvolupament com són: depurador de codi, biblioteca, simulador de telèfon basat en *QEMU*, documentació, exemples de codi i tutorials. Les plataformes de desenvolupament suportades *inclouen Linux, Mac OS X 10.4.9* o posterior i *Windows XP* o superior. La plataforma integral de desenvolupament (*IDE, Integrated Development Environment*) suportada oficialment és *Eclipse* juntament amb el complement *ADT* (*Android Development Tools plugin*), encara que també es pot utilitzar un editor de text per escriure fitxers Java i XML i utilitzar comandes en un terminal (es necessita els paquets *JDK, Java Development Kit* i *Apache Ant*) per crear i depurar aplicacions. I també pot controlar dispositius *Android* que estiguin connectats.

Les actualitzacions del *SDK* estan coordinades amb el desenvolupament general d'*Android*. El SDK també suporta versions antigues d' *Android*.

Una aplicació Android està composada per un conjunt de fitxers empaquetats en format .apk i guardada en el directori /data/app del sistema operatiu *Android* (aquest directori necessita permisos de superusuari, root, per raons de seguretat). Un paquet *APK* inclou fitxers .dex (executables *Dalvik*, el codi intermedi compilat específic d'*Android*), recursos, etc.[16]

5.1.1.5. IDE ECLIPSE KEPLER

Eclipse és un programa format per un conjunt d'eines de programació de codi obert multi plataforma per desenvolupar aplicacions de client. Aquesta plataforma s'ha utilitzat per desenvolupar entorns de desenvolupament integrats (*IDE*), com el *IDE* de *Java* que s'anomena *Java Development Toolkit (JDT)* i el compilador (*ECJ*) que s'entreguen com a part de Eclipse i són utilitzades per desenvolupar amb el mateix *Eclipse*. [15].

Per instal·lar *Eclipse* hem anat a la web i ens hem baixat el programa. Un cop baixat l'hem executat i s'ha creat on li hem indicat la estructura de directoris necessària. Al

iniciar el programa per primer cop ens demana la localització de un *workspace* un cop indicat ja es pot treballar amb aquesta aplicació.

5.1.1.6. ADT Android per Eclipse

ADT (Android Development Tools) és un plugin pel *IDE Eclipse* dissenyat per proporcionar al desenvolupadors un entorn integrat on construir aplicacions *Android*.

Per instal·lar aquest plugin com a requisits mínims es necessita:

- *Eclipse 3.7.3 (Indigo)* o superior.
- *Eclipse JDT plugin* (inclòs en la majoria de paquets del *Eclipse IDE*)
- *JDK 6 (JRE* no és suficient)
- No és compatible amb *GNU Compiler* per *Java*.

Podeu trobar els detalls d'instal·lació del *plugin* en el Annex 2 de la memòria.

5.1.1.7. API Google Maps

Un *API*, una interfície de programació d'aplicacions, és el conjunt de funcions i procediments (o mètodes, en la programació orientada a objectes) que ofereix certa biblioteca per ser utilitzat per un altre software com una capa d'abstracció.

Google Maps és un servidor d'aplicacions de mapes en la web que pertany a *Google*. Ofereix imatges de mapes, fotografies per satèl·lit del mon i inclús rutes entre diferents ubicacions.

API de *Google Maps* és una eina que no ve inclosa en la llibreria estàndard de *Android*, i tampoc es troba en el *SDK*. Per poder-la utilitzar és necessari afegir-la. Per fer-ho s'han de seguir una sèrie de passos descrits en l'Annex 3 d'aquesta memòria. [7]

5.1.2. Eines de control de versions

5.1.2.1. EGIT

Pel control de versions s'ha utilitzat un *plugin* de Eclipse anomenat *EGIT*. *EGIT* és un proveïdor per a *Eclipse* per sistema de control de versions *Git*.

Git és un SCM distribuït, que significa que cada desenvolupador te una copia completa de tota la història de cada revisió del codi, fen consultes contra la historia molt ràpid i de forma versàtil.

5.2. Desenvolupament de l'aplicació

En aquest apartat explicaré els aspectes rellevants que han afectat a la implementació del projecte.

He dividit el projecte en varis paquets:

- *com.pfc.platspreparats*, en aquest paquet bàsicament hem inclòs les classes que s'encarreguen de la part visual i les hem anomenat NompantallaActivity.java i les tasques asíncrones associades a les pantalles.

- *com.pfc.adaptadors*, en aquest paquet hem inclòs els adaptadors per omplir les dades de les pantalles i les classes Java dels diferents elements que utilitzem en la aplicació com són el plat, comanda i detall de la comanda.

- *com.pfc.librerias*, en aquest apartat hem inclòs les diferents classes que utilitzem per gravar dades en la base de dades *SQLite* del mòbil, per obtenir els *JSON* de Internet, per comprovar si hi ha connexió a Internet...

En aquest apartat explicaré els aspectes rellevants que han afectat a la implementació del projecte i ho enumeraré segons la tècnica utilitzada en el desenvolupament explicant si hi ha alguna raó per haver-ho fet amb aquesta tècnica enlloc d'utilitzar un altre mode d'implementar-ho.

5.2.1. Comunicació amb el servidor

Les opcions que dóna *Android* per comunicar-se amb el servidor son:

- ***AsyncTask (Tasques Asíncrones)***

  Aquestes tasques et permeten fer les tasques asíncrones de el interfície gràfic, operacions de bloqueig d'un *fil* secundari i publicar els resultats en el interfície gràfic.

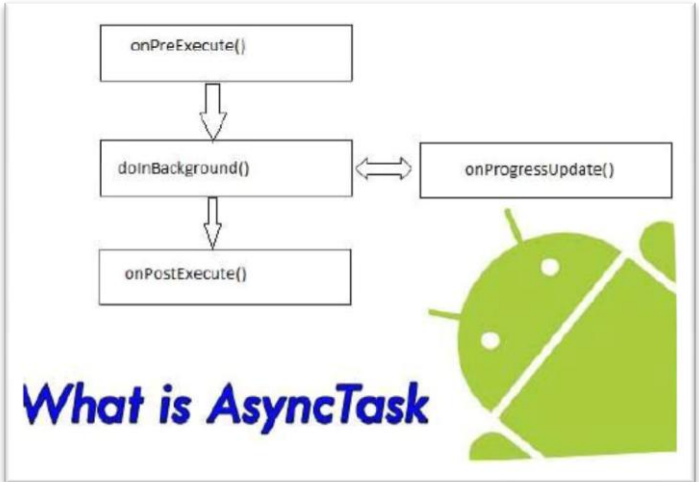

Figura 22: AsyncTask

- ***Services***

  Un *Service* és un component *Android* que viu independentment de qualsevol altre component. Les activitats poden anar i venir, però els serveis poden durar.

- ***Handler***

  Los *Handlers* implementen un mecanisme de pas de missatges entre fils de forma que ens permeten enviar missatges al nostre fil *UIThread*.

Ens hem decantat per utilitzar *AsyncTask* perquè utilitza un fil separat de la tasca principal de l'aplicació i pot realitzar processos en *background*, i considero que això és una avantatge i per aquesta raó l'he utilitzat en el desenvolupament de l'aplicació. Es més senzilla d'implementar que un *Handler*, i suficient per les nostres necessitats.

L'aplicació obté la informació dels plats del servidor, i s'ha fet la programació de tal forma que sigui la pantalla de Splash la que faci la petició de la informació dels plats que mantindrà la llista de plats actualitzada. Aquesta petició es fa en la pantalla d'entrada a l'aplicació per intentar minimitzar el nombre de vegades que es fa la petició.

Quan s'ha d'enviar informació a la base de dades de Internet s'envia la informació des de la tasca asíncrona associada a l'activitat, com és el cas de la comanda, la identificació, el registre, "*olvidé mi contraseña*"...

En tots aquests casos s'han utilitzat per la comunicació les tasques asíncrones (*AsyncTask*).

Aquí podem veure un el codi de la tasca asíncrona del registre d'un usuari:

```java
package com.pfc.platspreparats;

import org.json.JSONException;

public class RegisterTask extends AsyncTask<String, Void, Integer> {

    private ProgressDialog progressDialog;
    private RegisterActivity activity;
    private static String KEY_SUCCESS = "success";
    private static String KEY_UID = "uid";
    private static String KEY_NAME = "nombre";
    private static String KEY_TIPO = "tipo";
    private static String KEY_EMAIL = "email";
    private static String KEY_CREATED_AT = "created_at";
    private int responseCode = 0;

    /*Constructor that takes parameters passed by LoginFragment and stores them as class-
     * wide fields so that all methods can access necessary variables.
     * */
    public RegisterTask(RegisterActivity activity, ProgressDialog progressDialog)
    {
        this.activity = activity;
        this.progressDialog = progressDialog;
    }

    /*A necessary but very simple method that launches a ProgressDialog to show the
     * user that a background task is operating (registration).*/
    @Override
    protected void onPreExecute()
    {
        progressDialog.show();
    }
```

Figura 23: RegisterTask.java(I)

En la pantalla superior es pot veure la definició de la classe i el constructor d'aquest i el mètode *onPreExecute()* que mostra el *progressDialog*.

En aquesta figura inferiors es pot veure que s'agafen les dades de la pantalla i s'envien al servidor a través de un objecte *JSON*. Es comprova que el procés ha acabat correctament i el *JSON* amb les dades de retorn, si és correcte s'emmagatzemen en la base de dades del mòbil.

```
protected Integer doInBackground(String... arg0) {
    EditText userName = (EditText) activity.findViewById(R.id.REGISTRO_usuario);
    EditText nameEdit = (EditText) activity.findViewById(R.id.REGISTRO_nombre);
    EditText passwordEdit = (EditText) activity.findViewById(R.id.registro_passwd_1);
    String nombre = nameEdit.getText().toString();
    String email = userName.getText().toString();
    String password = Encriptar.encriptaEnMD5(passwordEdit.getText().toString());
    Log.v(email, password);
    Funciones funcion = new Funciones();
    JSONObject json = funcion.registerUser(nombre, email, password);
    try {
        if (json.getString(KEY_SUCCESS) != null) {
            //registerErrorMsg.setText("");
            String res = json.getString(KEY_SUCCESS);
            if(Integer.parseInt(res) == 1){
                // user successfully registred
                // Store user details in SQLite Database
                SQLiteHandler db = new SQLiteHandler(activity.getApplicationContext());
                JSONObject json_user = json.getJSONObject("usuario");
                // Clear all previous data in database
                Funciones.logoutUser(activity.getApplicationContext());
                db.addUser(json_user.getString(KEY_NAME), json_user.getString(KEY_EMAIL),json.getString(KEY_UID),
                        json_user.getString(KEY_TIPO) , json_user.getString(KEY_CREATED_AT));
                responseCode = 1;
            }else{
                // Error in registration
                responseCode = 0;
            }
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return responseCode;
}
```

Figura 24: RegisterTask (II)

Finalment en el mètode *onPostExecute()* el que es fa és segons si el *JSON* ha indicat que el procés ha finalitzat correctament es deixa la pantalla en blanc i es crida al procés de identificació i si ha arribat incorrecte es queda en la pantalla de registre i envia un error perquè el vegi l'usuari.

```
    /*This final method concludes the background task. Its responseCode variable is sent from
     * doInBackground, and this method acts based on the code it is sent. If the code is 1,
     * registration was successful and the main activity notifies the user of succes - the
     * inverse occurs if there is a failure and 0 was sent.*/
    @Override
    protected void onPostExecute(Integer responseCode)
    {
        EditText userName = (EditText)activity.findViewById(R.id.REGISTRO_usuario);
        EditText passwordEdit = (EditText)activity.findViewById(R.id.registro_passwd_1);

        if (responseCode == 1) {
            progressDialog.dismiss();
            activity.registerReport(responseCode);
            userName.setText("");
            passwordEdit.setText("");
            Intent i = new Intent();
            i.setClass(activity.getApplicationContext(), LoginActivity.class);
            activity.startActivity(i);

        }
        if (responseCode == 0) {
            progressDialog.dismiss();
            activity.registerReport(responseCode);
        }
    }
}
```

Figura 25: RegisterTask (III)

Hi ha un únic cas on he utilitzat el Service, proporcionat per Android, que es diu *DownloadManager*, per comunicar-me amb el servidor, i ha estat en la baixada dels fitxers de fotos del servidor al mòbil. I la raó per la que he triat aquesta opció en aquest cas és perquè vaig pensar que si els fitxers de fotos eren molt grans podia donar problemes i vaig preferir-los baixar amb aquest procés.

El *DownloadManager* és un servei del sistema que s'encarrega de descarregues *HTTP* de llarga durada. Els clients poden sol·licitar des de quin *URI* volen descarregar l'arxiu i en quin lloc el volen guardar. Aquest sistema durà a terme la descarrega en un segon pla, i els controls de errors de descarrega i reintents els fa el sistema.

```
//BAJAMOS EL FICHERO CON UN DOWNLOADMANAGER
File folder1 = new File( android.os.Environment.getExternalStorageDirectory().getAbsolutePath() + File.separator + "platspreparats" + File.separator
        + "img" + File.separator + plato.getString(KEY_FICHERO));
if (folder1.exists() ) {}
else {
    activity.getApplicationContext();
    dm = (DownloadManager)activity.getApplicationContext().getSystemService(Context.DOWNLOAD_SERVICE);
    Uri Download_Uri = Uri.parse("http://www.platspreparats.net/platspreparats/img/" + plato.getString(KEY_FICHERO));
    DownloadManager.Request request = new DownloadManager.Request(Download_Uri);
    File ruta = new File(
            android.os.Environment
            .getExternalStorageDirectory()
            + File.separator + "platspreparats" );

    if (!ruta.exists()) {
        ruta.mkdirs();
    }
    ruta = new File(
            android.os.Environment
            .getExternalStorageDirectory()
            + File.separator + "platspreparats" + File.separator + "img");

    if (!ruta.exists()) {
        ruta.mkdirs();
    }
    request.setDestinationInExternalPublicDir( File.separator + "platspreparats" + File.separator + "img" + File.separator ,  plato.getString(KEY_FICHERO));
    long enqueue = dm.enqueue(request);
}
```

Figura 26: Ús DownloadManager

5.2.2. Emmagatzemar dades persistents en Android

Les opcions que ens dóna Android per emmagatzemar dades persistentment són les següents:

- *Shared Preferences*: Permet emmagatzemar de forma privada dades primitives en parelles clau-valor.
- Emmagatzemament intern: Permet emmagatzemar de forma privada en la memòria del dispositiu.
- Emmagatzemament extern: Permet emmagatzemar de forma pública dades en la memòria externa compartida.
- Bases de dades *SQLite*: Emmagatzemament de dades estructurades en una base de dades privada.
- Connexió de xarxa: Emmagatzemament de dades en la web amb el teu propi servidor de xarxa.

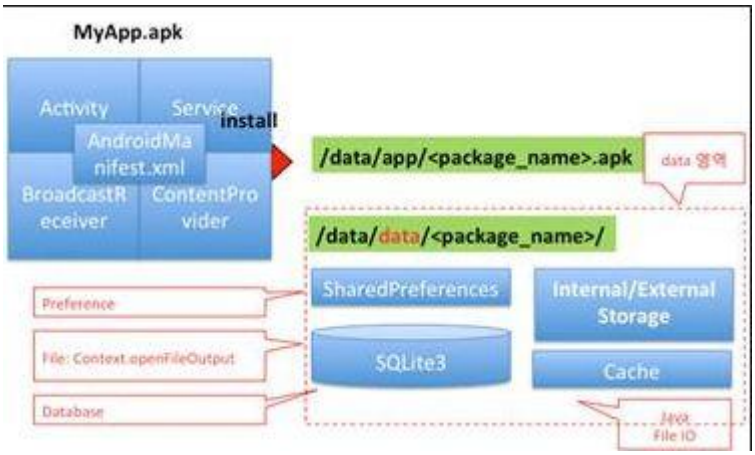

Figura 27: Emmagatzemament Android

En el desenvolupament de l'aplicació s'han utilitzat varis d'aquests mètodes d'emmagatzemar dades, a continuació indiquem com i perquè s'ha utilitzat cadascun d'ells.

En el cas de les *Shared Preferencies* les hem utilitzat en la pantalla de configuració on el usuari pot triar la manera de rebre les notificacions (encara que a causa de problemes de temps aquestes no s'han pogut implementar en el projecte que es presenta)

Per utilitzar aquest tipus d'emmagatzemament s'ha de crear un xml amb les opcions que es volen que surtin en la pantalla i guardar-lo dins del directori res\xml\ i amb al codi indicat a continuació és com s'utilitzen aquestes preferències.

```java
package com.pfc.platspreparats;

import android.annotation.SuppressLint;

@SuppressLint("NewApi")
public class PreferenciasActivity extends FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Display the fragment as the main content.
        getFragmentManager().beginTransaction()
                .replace(android.R.id.content, new PreferenciasFragment())
                .commit();
    }
}
```

Figura 28: PreferenciasActivity

```java
package com.pfc.platspreparats;

import android.annotation.SuppressLint;

@SuppressLint("NewApi")
public class PreferenciasFragment extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Load the preferences from an XML resource
        addPreferencesFromResource(R.xml.preferencias);
    }

}
```

Figura 29: PreferenciasFragment

La pantalla de l'aplicació desenvolupada a través de Shared Preferences és la de configuració:

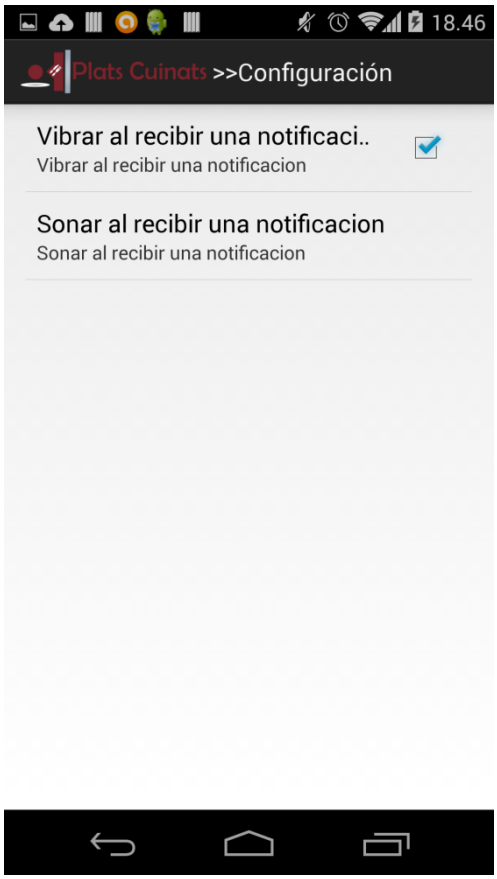

Figura 30: Pantalla Shared Preferences

També hem utilitzat l'emmagatzemament extern, que l'hem utilitzat per emmagatzemar els fitxers amb les fotografies que hem descarregat del servidor. Per poder utilitzar aquest tipus de emmagatzemament és necessari que el fitxer del *AndroidManifest.xml* tingui el permís corresponent per escriure en la targeta externa.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Figura 31: AndroidManifest.xml Emmagatzemament extern permisos

I per llegir els fitxers i mostrar les fotos per pantalla s'utilitza el següent codi:

```
if(plat != null) {
    // get the TextView from the ViewHolder and then set the text (item name) and tag (item ID) values
    v.plato.setText(plat.getPlato());
    v.precio.setText(plat.getPrecio());
    //a traves de fichero
    File f = new File(android.os.Environment
            .getExternalStorageDirectory()
            + File.separator + "platspreparats" + File.separator + "img" + File.separator + plat.getFichero());
    Bitmap bitmap;
    BitmapFactory.Options bitmapOptions = new BitmapFactory.Options();
    bitmapOptions.inJustDecodeBounds = true;
    bitmapOptions.inSampleSize = calculateInSampleSize(bitmapOptions, 100, 100);
    bitmapOptions.inJustDecodeBounds = false;

    bitmap = BitmapFactory.decodeFile(f.getAbsolutePath(), bitmapOptions);
    v.foto.setImageBitmap(bitmap);

}
```

Figura 32: Mostrar fotos

I també hem utilitzat la base de dades SQLite. A continuació es poden veure uns extractes de codi on es veu quina classe s'utilitza: la classe *SQLiteHandler*, que extends *SQLiteOpenHelper* i com es creen les taules i es poden consultar, modificar i eliminar.

```
package com.pfc.platspreparats.librerias;

import java.util.ArrayList;

public class SQLiteHandler extends SQLiteOpenHelper {

    // All Static variables
    // Database Version
    private static final int DATABASE_VERSION = 1;
```

Figura 33: Classe SQLiteHandler extends SQLiteOpenHelper

```
// Creating Tables
@Override
public void onCreate(SQLiteDatabase db) {
    String CREATE_LOGIN_TABLE = "CREATE TABLE " + TABLE_LOGIN + "("
            //+ KEY_UID + " TEXT PRIMARY KEY,"
            + KEY_EMAIL + " TEXT PRIMARY KEY,"
            + KEY_NAME + " TEXT,"
            + KEY_CREATED_AT + " TEXT" + ")";
    db.execSQL(CREATE_LOGIN_TABLE);
    onCreatePlato(db);
    onCreatePedido(db);
    onCreateDetalle(db);
}
```

Figura 34: Crear taules SQLite

```
public void addUser(String nombre, String email, String uid, String created_at) {
    SQLiteDatabase db = this.getWritableDatabase();

    ContentValues values = new ContentValues();
    //values.put(KEY_UID, uid); // Email
    values.put(KEY_EMAIL, email); // Email
    values.put(KEY_NAME, nombre); // Name
    values.put(KEY_CREATED_AT, created_at); // Created At

    // Inserting Row
    db.insert(TABLE_LOGIN, null, values);
    db.close(); // Closing database connection
}

public void crearPlato (String idplato, String tipo, String plato, String descripcion, String ingredientes,  String precio, String situ
    SQLiteDatabase db = getReadableDatabase();
    String[] valores_recuperar = {KEY_UIDPlato};
    Cursor c = db.query(TABLE_PLATO, valores_recuperar, KEY_UIDPlato +"="+ idplato,null, null, null, null, null);
    if(!c.isAfterLast()) {
        c.close();
        db.close();

        updatePlato(idplato, tipo, plato, descripcion, ingredientes, precio, situacion,  calorias, created_at, modified_at, Fichero);
    }
    else {
        c.close();
        db.close();
        addPlato(idplato, tipo, plato, descripcion, ingredientes, precio, situacion,  calorias, created_at, modified_at, Fichero);
    }
```

Figura 35: Afegir i consultar registres

I finalment també hem utilitzat emmagatzematge a través de la xarxa, tenint un servidor extern amb la seva base de dades, que és on es guarden totes les dades compartides per tots els usuaris perquè estiguin accessibles a les consultes corresponents. La comunicació es fa a través de *PHP* i a continuació es pot veure com ens connectem a la base de dades a través de *PHP* i algunes consultes realitzades pels programes.

Al servidor s'ha creat un arxiu *PHP* on s'indica la configuració de la base de dades (*db_config.php*) i un arxiu de connexió (*db_connect.php*) on s'ha configurat la connexió a partir de l'arxiu de configuració. I a part d'aquests dos arxius en tenim un de funcions (*DB_Functions.php*) on es realitza l'emmagatzemament i consulta a la base de dades i un últim arxiu a través del qual arriben totes les peticions (*index.php*) .

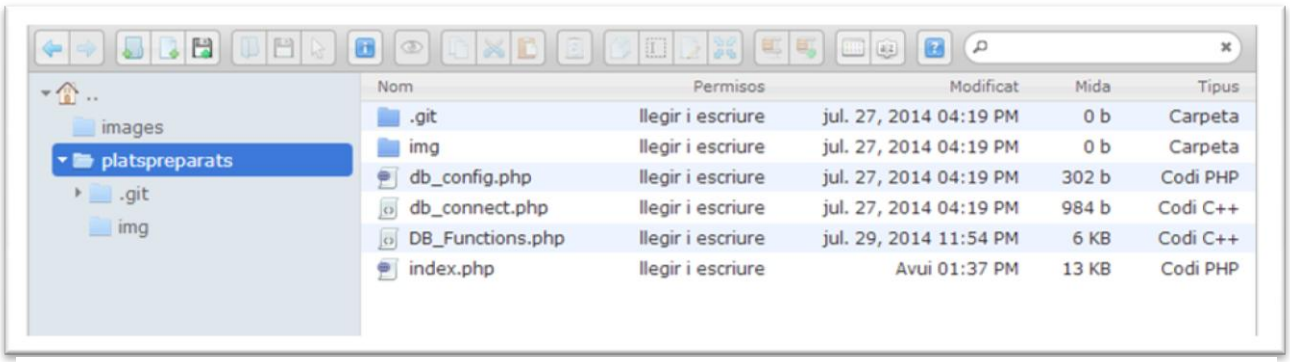

Figura 36: Estructura Arxius Servidor

5.2.3. Intents implícits

Un altre aspecte a tenir en compte en la programació i que volia utilitzar són les pròpies funcionalitats del telèfon i provar de programar-les en la part de la botiga. Així volia poder fer que al tocar el telèfon, es pogués trucar directament, al demanar la web s'obrís la pagina web, al tocar el mail es pogués enviar un mail directament. I també pogués utilitzar la càmera per obtenir les fotos del plat per pujar-les al servidor amb posterioritat. Totes aquestes accions queden resoltes amb l'ús dels Intents implícits.

Aquí es veu el codi de la part de la botiga quan esta cridant les accions del mòbil directament.

```
public class BotigaActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_botiga);
    }

    public void pgWeb(View view) {
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.platspreparats.net"));
        startActivity(intent);
    }

    public void llamadaMovil(View view) {
        Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:669326910"));
        startActivity(intent);
    }

    public void mandarCorreo(View view) {
        Intent intent = new Intent(Intent.ACTION_SEND);
        intent.setData(Uri.parse("mailto:"));
        intent.setType("text/plain");
        intent.putExtra(Intent.EXTRA_SUBJECT, "asunto");
        intent.putExtra(Intent.EXTRA_TEXT, "texto del correo");
        intent.putExtra(Intent.EXTRA_EMAIL,  "platspreparats@gmail.com");
        intent.putExtra(Intent.EXTRA_CC,  "platspreparats@gmail.com");
        startActivity(intent);
    }
```

Figura 37: Botiga.java

Un altre *indent implícit* és el que utilitzem per fer una foto des de l'aplicació i agafar aquesta imatge. Aquest es pot veure en el següent tros de codi:

```
private void selectImage() {

    final CharSequence[] options = { getString(R.string.hazfoto),
            getString(R.string.elige), getString(R.string.cancelar) };

    AlertDialog.Builder builder = new AlertDialog.Builder(NuevoPlatoActivity.this);
    builder.setTitle(getString(R.string.add_foto));
    builder.setItems(options, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int item) {
            if (options[item].equals(getString(R.string.hazfoto))) {
                Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
                File f = new File(android.os.Environment
                        .getExternalStorageDirectory(), "temp.jpg");
                intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(f));
                startActivityForResult(intent, 1);
            } else if (options[item].equals(getString(R.string.elige))) {
                Intent intent = new Intent(
                        Intent.ACTION_PICK,
                        android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
                startActivityForResult(intent, 2);

            } else if (options[item].equals(getString(R.string.cancelar))) {
                dialog.dismiss();
            }
        }
    });
    builder.show();
}
```

Figura 38: Intent implícit utilitzar càmera fotos mòbil

A part de posar la part de codi indicat, al *AndroidManifest.xml* s'han de indicar els permisos corresponents, com és en el cas d'utilitzar la càmera i emmagatzemar les fotos en la targeta SD, s'han d'afegir els permisos corresponents.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.pfc.platspreparats"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="19" />

    <permission
        android:name="org.example.ejemplogooglemaps.permission.MAPS_RECEIVE"
        android:protectionLevel="signature" />

    <uses-permission android:name="org.example.ejemplogooglemaps.permission.MAPS_RECEIVE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.CAMERA" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/plats500"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
```

Figura 39: AndroidManifest.xml

5.2.4. API Google Maps

També es volia provar l'ús d'un *API de Google* en l'aplicació i el que s'ha decidit provar és el *Google Maps* versió 2. Aquest procés la part que sembla més complicada en un inici és aconseguir correctament el *API Key* perquè entendre que demana exactament costa una mica d'entendre, però per sort, avui en dia en Internet hi ha molta informació (Annex 3).

Per utilitzar el *API de Google Maps* s'ha d'afegir al *AndroidManifest.xml* una sèrie de característiques descrites amb detall en el Annex 3 del projecte.

I aquí es pot veure el detall de funcionament del botó YO que el que fa és segons la posició indicada pel *GPS* del mòbil indicar on s'està situat sobre el mapa i traça la ruta entre la posició actual del usuari i la botiga.

```
public void animateCamera(View view) {
    LocationManager locManager = (LocationManager)getSystemService(LOCATION_SERVICE);
    if (!locManager.isProviderEnabled(LocationManager.GPS_PROVIDER)) {
        mostrarAvisoGpsDeshabilitado();
    }

    if (mapa.getMyLocation() != null)
    {
        LatLng diryo = new LatLng( mapa.getMyLocation().getLatitude(),mapa.getMyLocation().getLongitude());
        mapa.animateCamera(CameraUpdateFactory.newLatLngZoom(diryo, 15));
        mapa.addMarker(new MarkerOptions()
        .position(diryo)
        .title("Yo")
        .snippet("Yo")
        .icon(BitmapDescriptorFactory
                .fromResource(R.drawable.ic_action_yo))
                .anchor(0.5f, 0.5f));
        markerPoints.add(diryo);
        if(markerPoints.size() >= 2){
            LatLng origin = markerPoints.get(0);
            LatLng dest = markerPoints.get(1);

            // Getting URL to the Google Directions API
            String url = getDirectionsUrl(origin, dest);

            DownloadTask downloadTask = new DownloadTask();

            // Start downloading json data from Google Directions API
            downloadTask.execute(url);
        }
    }
}
```

Figura 40: Boto YO mapa

5.2.5. Altres detalls de la implementació

Altres detalls que poden interessar de la implementació són els següents:

- ***Seguretat de l'usuari***

En quan a la part de seguretat de l'usuari s'ha decidit encriptar la contrasenya del usuari abans d'enviar-la al servidor web, i s'ha fet a través de la classe Encriptar:

```
package com.prc.piatspreparats.llibreries;

import java.security.MessageDigest;

public class Encriptar {
    private static final char[] CONSTS_HEX = { '0', '1', '2', '3', '4', '5',
        '6', '7', '8', '9', 'a', 'b', 'c', 'd', 'e', 'f' };

    public static String encriptaEnMD5(String stringAEncriptar) {
        try
        {
            MessageDigest msgd = MessageDigest.getInstance("MD5");
            byte[] bytes = msgd.digest(stringAEncriptar.getBytes());
            StringBuilder strbCadenaMD5 = new StringBuilder(2 * bytes.length);
            for (int i = 0; i < bytes.length; i++)
            {
                int bajo = bytes[i] & 0x0f;
                int alto = (bytes[i] & 0xf0) >> 4;
            strbCadenaMD5.append(CONSTS_HEX[alto]);
            strbCadenaMD5.append(CONSTS_HEX[bajo]);
            }
            return strbCadenaMD5.toString();
        }
        catch (NoSuchAlgorithmException e)
        {
            return null;
        }
    }
}
```

Figura 41: Classe Encriptar

- ***Connexió a Internet.***

En l'aplicació s'ha creat una classe per poder comprovar si existeix connexió abans de intentar enviar les dades al servidor, per intentar evitar errors de connexió inesperats. La classe que controla aquest tema es la següent:

```
package com.pfc.platspreparats.librerias;
import android.content.Context;

public class ConexionInternet {

    private Context contexto;
    public ConexionInternet(Context contexto)
    {
        this.contexto = contexto;
    }
    public boolean estasConectado()
    {
        ConnectivityManager conectividad = (ConnectivityManager)
                contexto.getSystemService(Context.CONNECTIVITY_SERVICE);
        if (conectividad != null)
        {
            NetworkInfo[] informacion = conectividad.getAllNetworkInfo();

            if(informacion != null)
                for (int i= 0; i< informacion.length; i++)
                    if (informacion[i].getState() == NetworkInfo.State.CONNECTED)
                        return true;
        }// fin de if conectividad
        return false;
    }
}
```

Figura 42: Classe connexió a internet

- ***Llistes de plats***

En l'aplicació tenim dos llistes de plats una que s'utilitza en el menú de l'usuari i l'altra es la utilitzada per l'administrador i te una sèrie de detalls de la implementació que m'agradaria incloure en aquesta secció.

La llista de plats de l'usuari, es un *ListView* on hi ha una imatge, dos *TextView* i dos botons que permeten incloure un plat a la comanda o traure'l segons convingui a l'usuari.

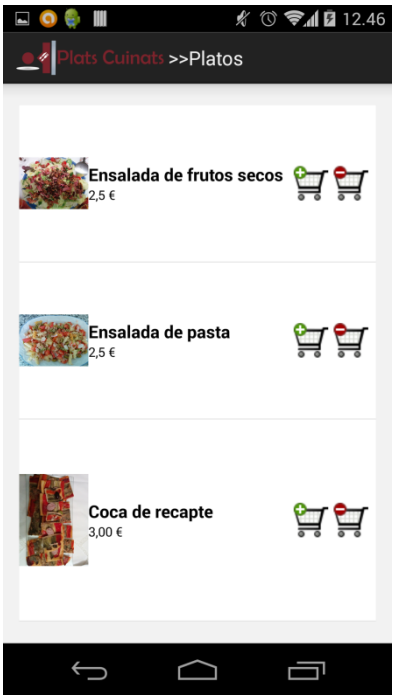

Figura 43: Pantalla plats usuari

En quan a la llista de plats pròpia de l'administrador, consisteix en un *ListView*, un *TextView*, dos *Checkbox* i un botó.

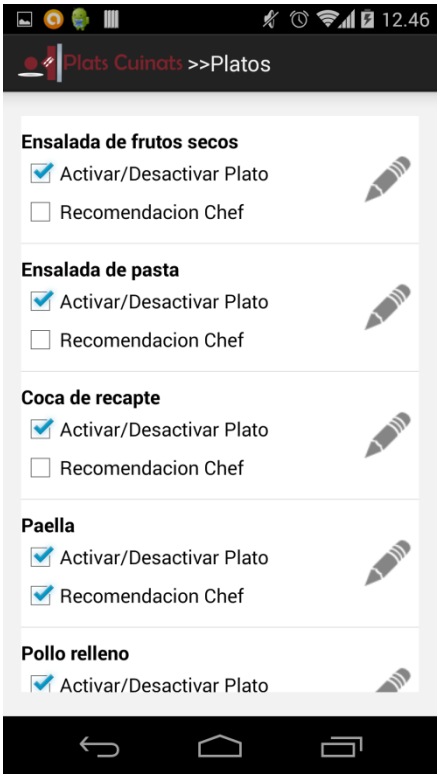

Figura 44: Llista plats administrador

En els dos casos quan es selecciona qualsevol element de una de les dues llistes s'obre la pantalla de la fitxa dels plats. Però en *Android*, per defecte, només permet tenir un element seleccionable, això vol dir que si posem un *CheckBox* en el *ListView* si seleccionem el element de la llista, aquesta no mostrarà la fitxa del plat. Però es pot fer,

a través de codi, que en la llista hi hagi més d'un element seleccionable, com en la llista de plats de l'administrador. Això ho aconseguim de la següent manera, anem al xml de la pantalla de la llista de plats de l'administrador i apliquem l'atribut *android:focuscable = "false"*. Amb aquesta modificació es permet que el *CheckBox* i l'element de la llista siguin seleccionables. A continuació podem veure el xml de la pantalla on s'ha aplicat aquesta propietat.

```
<LinearLayout
    android:id="@+id/linlay"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/Plato"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:singleLine="false"
        android:text="Large Text"
        android:textAppearance="?android:attr/textAppearance"
        android:textStyle="bold" />

    <CheckBox
        android:id="@+id/actdes"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@+id/Plato"
        android:clickable="false"
        android:focusable="false"
        android:text="Activar/Desactivar Plato" />

    <CheckBox
        android:id="@+id/recomen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@+id/Precio"
        android:clickable="false"
        android:focusable="false"
        android:text="Recomendacion Chef" />
```

Figura 45: xml pantalla usuari administrador

5.3. Dificultats trobades en la implementació

5.3.1. Blobs

Un *BLOB (Binary Large Objects*) són elements utilitzats en les bases de dades per emmagatzemar objectes de mida gran que canvien de forma dinàmica. Generalment, aquestes dades són imatges, arxius de so i altres objectes multimèdia.

Inicialment la implementació de l'aplicació es basava en els *blobs* per fer la càrrega i descàrrega de fotos des del servidor, així com el tractament de una foto perquè en cada pantalla on s'utilitza la mida s'adaptés a la que li convenia en la pantalla. Però després de molts intents i de no saber on hi havia el problema es va decidir fer un canvi en la implementació i utilitzar intercanvi de fitxers, tal i com s'ha presentat anteriorment.

5.4.Test realitzats

Al donar per finalitzada la implementació de l'aplicació i per comprovar el funcionament de la mateixa en real s'ha decidit realitzar un test informal que ha consistit en demanar a diferents usuaris de mòbils Android i tablets si volien realitzar un test a una nova aplicació i respondre a unes preguntes i aportar les seves opinions sobre l'aplicació.

Al final hem aconseguit 5 voluntaris diferents, amb mòbils diferents. I les preguntes que s'han realitzat son les següents:

- Que t'ha semblat el disseny de l'aplicació?
- Creus que si l'aplicació estigués disponible a el *Google Play* i tinguessis la botiga prop de la feina o casa utilitzaries l'aplicació?
- Que es el que t'ha agradat menys de l'aplicació?
- Has tingut algun error o algun problema de funcionament amb l'aplicació?
- Has trobat a faltar alguna cosa per fer mes entenedor l'ús de l'aplicació?
- Que canviaries de l'aplicació per fer-la mes al teu gust?

Les conclusions a les quals s'ha arribat després de conèixer les opinions del usuaris que han provat l'app són les següents:

- Als usuaris els hi ha semblat un entorn net.
- L'eina els ha semblat útil i prou innovadora ja que no coneixien cap aplicació especialitzada en el tema, encara que sí coneixien eines de demanar menjar a restaurants de diferents tipus.
- Les proves que han realitzat de registrar-se, accedir, consultar els plats i fer una comanda els hi ha semblat satisfactòries.
- Han demanat millorar la navegació d'algunes pantalles. Al inici quan s'obre l'aplicació i s'entra directament a la pantalla de la llista de plats, no hi ha cap menú i en un principi a algun usuari li ha estat difícil trobar que amb la tecla de retrocés s'arriba a la pantalla principal de menú.
- També han indicat que a la pantalla de Fer Comanda quan s'envia la comanda no estaven segurs si havia arribat o no, perquè desapareixien les dades i no sortia cap missatge de comanda rebuda.
- Millorar la pantalla de la comanda, afegint un import total per saber quan puja la comanda.

- També s'ha provat en tablets i encara que les pantalles no estan ajustades exactament a la mida d'aquests, l'aplicació també funciona. En la imatge següent es pot veure la pantalla de la llista de plats vista des d'un tablet.

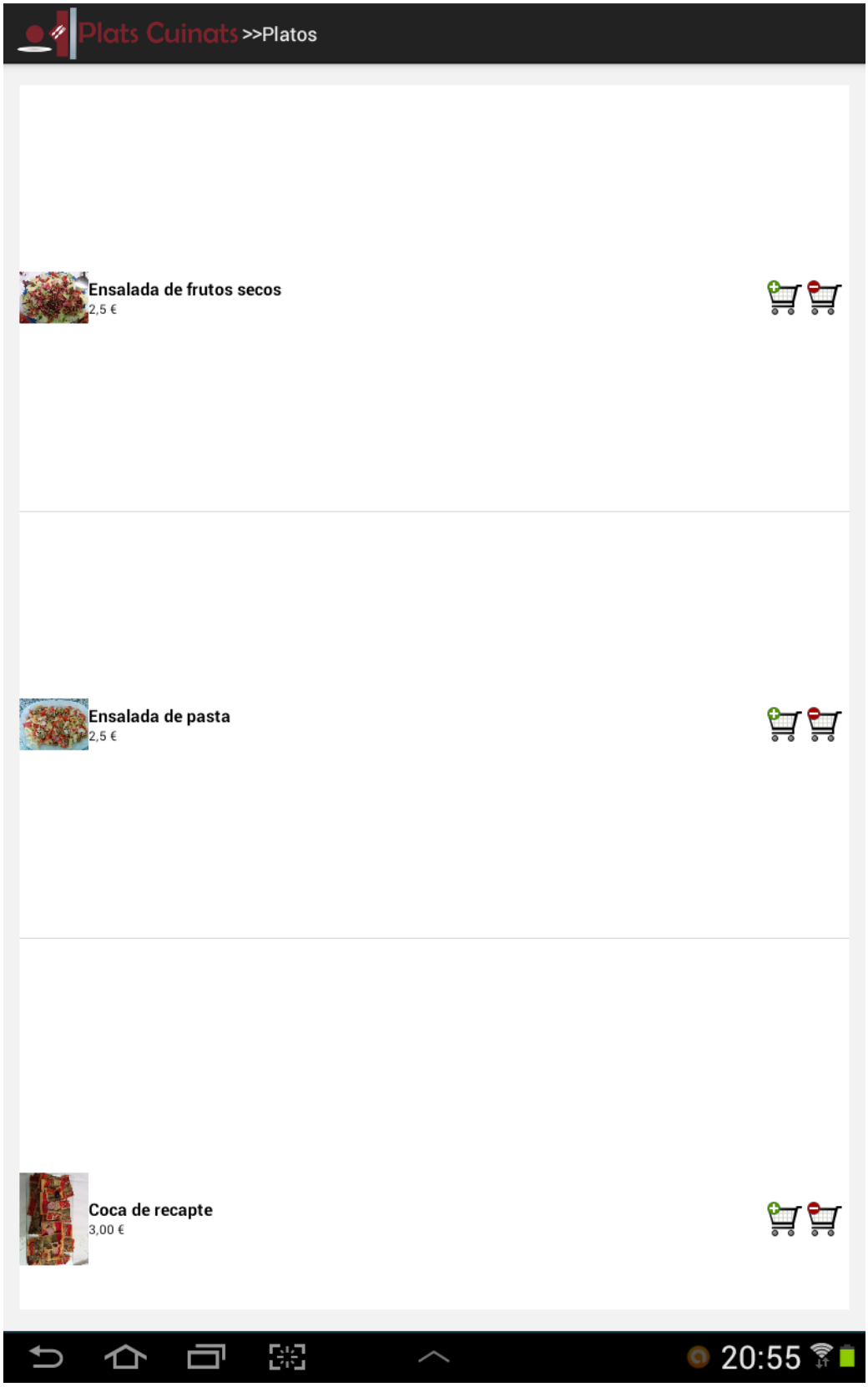

Figura 46: Pantalla llista plats amb tablet

- S’ha provat amb diferents versions d’*Android*, des de 2.3 fins a la 4.4 i sembla que en les versions provades funciona.

- En algun mòbil han tingut problemes en la utilització del mapa de l’aplicació, però segons indicacions del usuari, tampoc li funcionen altres aplicacions que tinguin relació amb el *Google Maps*.

- S’han demanat que es posin més missatges per informar millor a l’usuari de les accions realitzades.

- En un Samsung GT-I9100 el menú lateral es queda enganxat moltes vegades.

Els dispositius mòbils amb els que s’han realitzat les proves i les versions Android que tenen instal·lades son:

- BQ Aquarios 5.0 – versió *Android* 4.2.1. (problema amb *Google Maps*)

- Samsung GT-I9100 – versió *Android* 4.1.2 (problema amb menú lateral)

- Nexus 5 – versió *Android* 4.4.4

- Sony Ericsson X10 – versio *Android* 2.3.1

- Tablet Samsung Galaxy Tab 2 10.1 – versió *Android* 4.0.3

Capítol 6: Conclusions i treball futur

M'agradaria destacar de l'aplicació, que després de fer una cerca per *Google Play*, que és una aplicació bastant innovadora, que explícitament de botigues de menjars preparats, que no siguin restaurants, no n'he trobat i respecte a les que ofereixen menjar a domicili és bastant similar a la resta. Un dels aspectes que més m'agrada de l'app desenvolupada, és que són dos aplicacions en una, la part de l'usuari i la part de l'administrador tot en una aplicació. I el principal problema es que no s'ha pogut deixar totalment acabada i l'aplicació en aquest moment no és multiempresa.

6.1. Conclusions

La realització d'aquest projecte ha estat més complicada del que havia previst en un moment inicial, a causa d'haver de compaginar una feina a temps complet i la realització del projecte, amb tota la preparació i feina prèvia d'aprenentatge dels diferents llenguatges i eines que s'han utilitzat, ja que no tenia coneixement de cap d'ells, màxim coneixements del llenguatge *SQL*, però no havia treballat ni amb *PHP*, *Java*, *MySQL* anteriorment.

El que tenia molt clar és que volia realitzar una aplicació per una plataforma mòbil. La plataforma elegida ha estat Android. Hem elegit aquesta plataforma en primer lloc, perquè els dispositius que tenia més a l'abast són Android, en segon lloc perquè estic a favor de la filosofia *Open Source* i en tercer lloc perquè *Android* és la segona plataforma mòbil amb nombre d'usuaris, cada cop més utilitzada. Per obtenir dades mes concretes es pot consultar http://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2014/.

Quant a la decisió de l'aplicació a realitzar, a causa d'un cop de sort, uns amics van muntar una botiga de plats cuinats i al començar ells amb el projecte i per donar-los un valor afegit els hi vaig proposar de fer l'aplicació. Aleshores ens vam plantejar que és el que creien que més els hi convenia, i vam veure que el que necessitaven era donar-se a conèixer, per aquesta raó es va fer que a l'aplicació pogués entrar tothom a veure els plats i la fitxa d'aquests i que només fos necessari identificar-se per poder enviar les comandes a través del mòbil, per la resta d'utilitats de l'aplicació com es informar de l'horari de la botiga, adreça, telèfon, mail, web ... no es necessites.

També es va voler integrar en l'aplicació la utilització d'un *API*, en el nostre cas el *API* del *Google Maps*. La dificultat principal de fer això en l'aplicació és entendre com funciona el protocol i com adaptar-lo per integrar-lo en la nostra aplicació Android ja que no és transparent fer-ho, i entendre com fer-ho és complicat. Per sort, avui en dia hi ha molta informació a l'abast a través de la web.

A nivell personal estic molt satisfeta per tot el procés i per tot el què he aprés durant el procés, i encara que m'hagués agradat una aplicació millor acabada, per falta de temps no ha pogut ser. Encara que el nivell de coneixements que he adquirit ha estat molt elevat i el meu nivell en aquest moment no te res a veure amb el que tenia al inici del projecte.

6.2. Treball futur

- Millorar els accessos a Internet i el tràfic d'Internet de l'aplicació controlant millor els accessos al servidor de forma que no es facin peticions innecessàries.

- Millorar el control de errors i intentar evitar que l'aplicació es quedi penjada amb un error inintel·ligible.

- Millorar el tractament d'imatges, ja que inicialment es volia fer amb l'ús de *blobs* emmagatzemant la imatge a les bases de dades i mostrar-la en diferent formats quan es crides a l'aplicació, però no hem aconseguit muntar-ho d'aquesta manera, i hem acabat guardant el nom del fitxer i mostrant la imatge a través del fitxer.

- També inicialment es volia incorporar al projecte l'enviament de notificacions conforme la comanda ja està preparada a la botiga i el client la pot passar a buscar sense tenir que esperar, però a causa de no tenir prou temps aquesta funcionalitat no s'ha incorporat en aquesta versió.

- Permetre a l'usuari aportar les seves opinions sobre l'aplicació, i permetre que fessin suggeriments i valoressin els plats i inclús podrien interactuar entre ells i amb el cuiner per comentar els plats i que es podria millorar d'ells o quins creuen que seria bo que s'oferissin a la botiga. I amb visibilitat a les diferents xarxes socials.

- Fer un estudi d'usabilitat i accessibilitat, per millorar l'ús per part de l'usuari i fer les modificacions necessàries per persones amb dificultats tan de vista com tàctils, poden incorporar la petició de comandes a través de l'accés a través de la parla.

- En un futur també es podria plantejar oferir repartiment a domicili i fer el pagament de les comandes a través del mòbil. I fer que l'aplicació sigui multiempresa.

- Internacionalitzar l'eina per permetre elegir el idioma amb que és vol utilitzar.

- Categoritzar els plats, per millorar la selecció d'aquests i permetre fer cerques sobre les llistes.

- Incorporar un mòdul per fer ofertes, descomptes i promocions de diversos tipus

- Oferir la possibilitat de fer comandes periòdiques, demanar càterings per festes especials.

Bibliografia

[1] Llenguatge PHP: http://es.wikipedia.org/wiki/PHP

[1] Julie C. Meloni. *PHP, MySQL y APACHE (PROGRAMACION).* Ed. Anaya Multimedia, 2009.

[2] Servidor Apache: http://es.wikipedia.org/wiki/Servidor_HTTP_Apache

[3] MySQL: http://es.wikipedia.org/wiki/MySQL

[4] Android: http://www.androidcurso.com/index.php/99

[5] Android: http://www.elandroidelibre.com/2014/02/aprende-android-en-20-conceptos-conceptos-1-y-2.html

[6] Disseny interficie: http://www.ecured.cu/index.php/Dise%C3%B1o_de_Interfaces_de_Usuario

[7] Android: http://www.sgoliver.net/

[8] Android: http://developer.android.com/guide/topics/ui/index.html

[9] Android: http://commonsware.com/Android/

[10] SDK Android: http://developer.android.com/sdk/index.html

[11] IDE Eclipse: http://www.eclipse.org/downloads/

[12] Postman REST-Client: https://chrome.google.com/webstore/detail/postman-rest-client/fdmmgilgnpjigdojojpjoooidkmcomcm

[13] Diagrames base de dades: http://diagrams.seaquail.net/

[14] PHPMyAdmin: http://es.wikipedia.org/wiki/PhpMyAdmin

[15] Eclipse: http://es.wikipedia.org/wiki/Eclipse_(software)

[16] Android SDK: http://es.wikipedia.org/wiki/Desarrollo_de_programas_para_Android

[17] Sea Quail Database Schema Tool https://chrome.google.com/webstore/detail/sea-quail-database-diagra/elkpialiknkiaebieojbgnhindepnlkg/details

[18] Model MVC http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

[19] Mark L. Murphy. *The Busy Coder's Guide to Android Development*

[20] Diagrama de classes: http://www.objectaid.com/home

[21] Jesús Tomás Gironés. *El Gran Libro de Android.* Ed. Marcomo.

[22] Jesús Tomás Gironés. *El gran libro de Android Avanzado.* Ed. Marcomo.

[23] Cay S. Horstmann – Gary Cornell. *Core Java Volumen I y II*. Ed. Pearson Prentice Hall.

[24] Bruce Eckel. *Piensa en Java 4/e.* Ed. Pearson Prentice Hall.

Annex 1: Manual d'usuari.

A1.1. Splash

L'aplicació comença amb la pantalla de Splash consistent en el logo de l'aplicació. Aquesta pantalla permet la consulta i actualització de les dades dels plats.

Figura 47: Pantalla Splash

A1.2. Llista de plats

Al entrar a l'aplicació apareix la llista de plats. A partir d'aquesta llista es poden fer vàries accions:

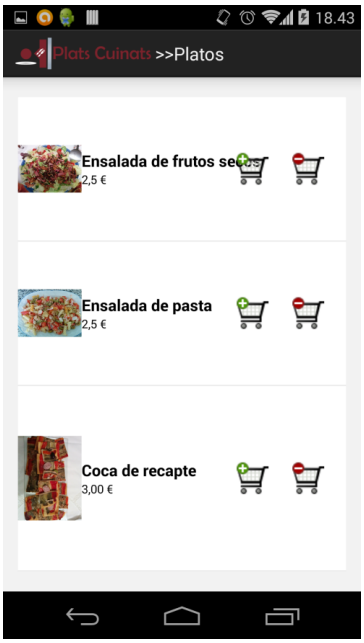

Figura 48: Pantalla Llista de plats

En aquesta llista hi ha 2 imatges d'un carret de compra amb un signe més i un amb un signe menys:

El  mira si existeix una comanda amb el plat indicat, si existeix afegeix +1 a la quantitat de la comanda d'aquell plat, sino existeix la crea amb quantitat 1.

El  revisa mira si existeix una comanda amb el plat indicat, si existeix resta -1 a la quantitat de la comanda d'aquell plat

A1.3. Fitxa de plats

Seleccionant sobre un element de la llista apareix la fitxa del plat.

Figura 49: Fitxa de plats

A1.4. Menús

A l'aplicació hi ha 2 menús desde menú lateral l'usuari pot accedir als menus Plats,

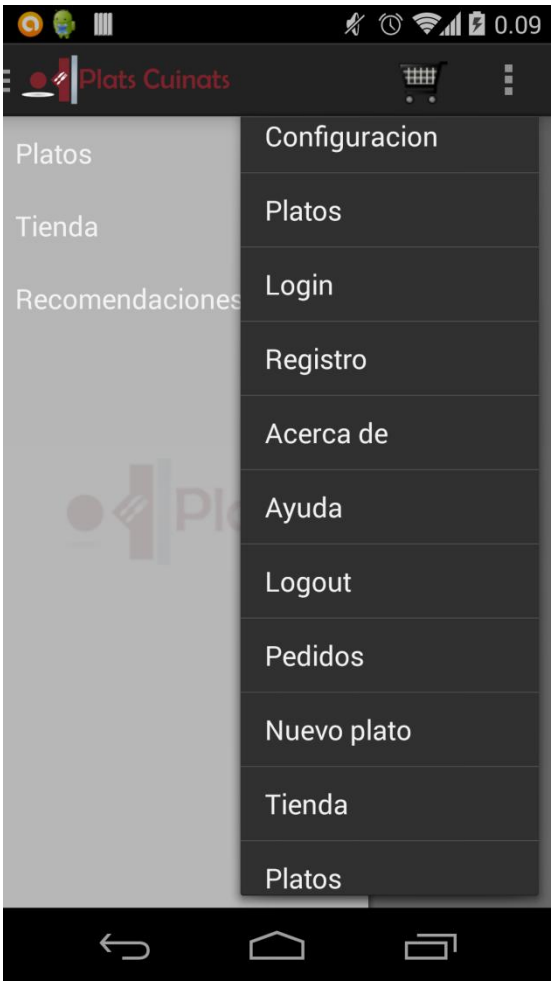

Figura 50: Menús

Botiga i Recomanacions. El menú Plats ja l'hem explicat abans.

Els menús, Pedidos, Nuevo Plato y Platos només estan disponibles per l'administrador.

A1.5. Botiga

El menú botiga ens porta a la pantalla de la botiga on es veu el següent:

Figura 51: Botiga

Aquesta pantalla permet fer el següent:

Seleccionant sobre l'adreça anem a parar a la pantalla de mapa, on ens dona la situació de la botiga, marcada amb la icona de la botiga.

Figura 52: Mapa

En aquesta pantalla hi ha 3 botons a la part inferior.

Figura 53: Botons mapa

Tienda:

Seleccionant sobre aquest botó es provoca que el mapa es centri en la botiga.

Yo:

Seleccionant sobre aquest botó i si el GPS està activat, fa que el mapa es centri sobre la posició en que estas situat tu en aquell moment, i a més, es mostra la ruta per arribar fins a la botiga.

Figura 54: Mapa amb el boto YO

Marcador:

Seleccionant sobre aquest botó es queda marcat amb un marcador el punt on has seleccionat.

Figura 55: Mapa amb el boto Marcador

Seleccionant sobre el telèfon l'aplicació et permet fer la trucada de telèfon directament.

Figura 56: Telefonar

Seleccionant sobre l'adreça de mail et permet obrir l'aplicació per enviar un mail a la botiga.

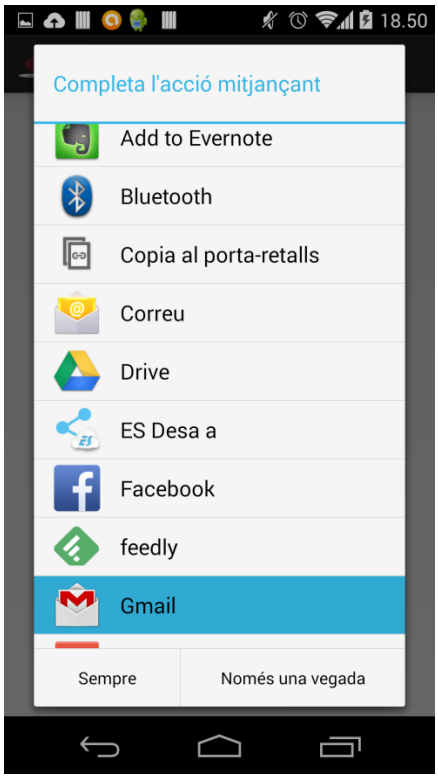

Figura 57: Enviar email

Seleccionant sobre l'adreça de la web et permet accedir a la web de l'empresa.

Figura 58: Web corporativa

A1.6. Recomanació Xef

El següent menú de Recomendaciones accedeix directament a la fitxa del plat recomanat.

Figura 59: Recomanació Xef.

A1.7. Configuració

En el menú superior dret hi ha aquestes opcions de menú més la resta d'opcions disponibles.

Configuració: permet configurar el so/vibració al rebre una notificació

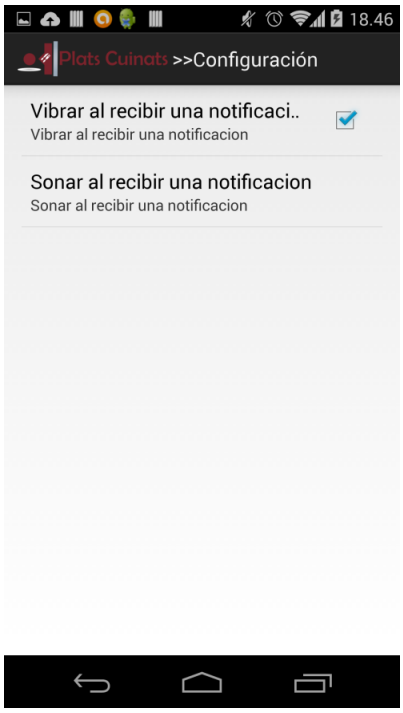

Figura 60: Configuració

Si es vol, que el telefon vibri al rebre una notificació s'ha de marcar el check box de la primera línea.

Si es vol elegir un so, diferent al predeterminat, al rebre una notificació seleccionant sobre la segona línia apareix una pantalla on seleccionar el so del mòbil, cap o el so de l'aplicació.

A1.8. Acerca De

Acerca de: Mostra la pantalla amb les dades de la versió i el copyright

Figura 61: Acerca De

A1.9. Ajuda

Ajuda: apareix una pantalla on hi ha una explicació del funcionament de l'aplicació.

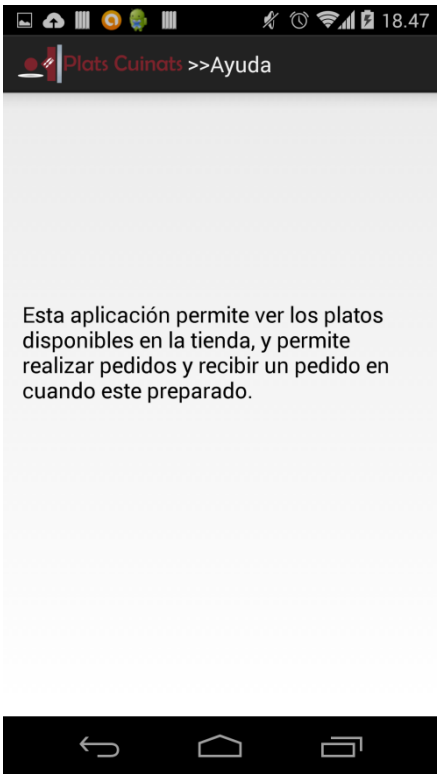

Figura 62: Ajuda

A1.10. Login

Login: pantalla que et permet accedir a l'aplicació, necessari per poder fer comandes i rebre les notificacions.

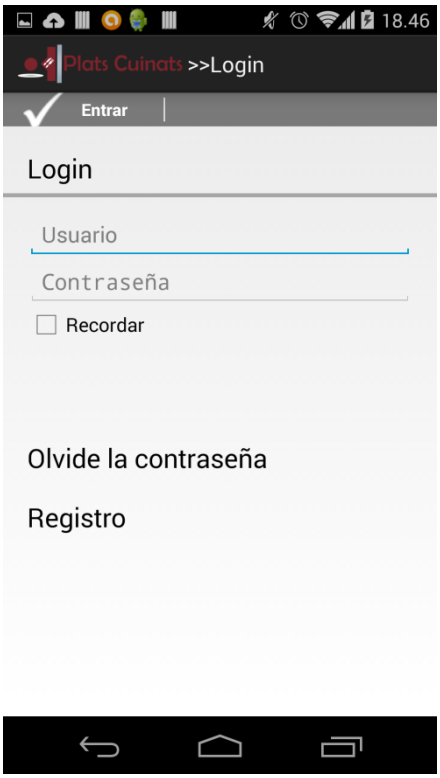

Figura 63: Identificació

En el camp Usuario, s'ha d'introduir el email utilitzar per registrarse. En el camp Contraseña s'ha d'introduir la contrasenya utilitzada en el registre. Si es vol que l'usuari es guardi en el telèfon i no tenir que tornar a Identificar-se en aquest mòbil s'ha de marcar el tick al costat de Recordar.

Aquesta pantalla permet anar a la pantalla per registrarse seleccionant en el Registro.

I si no es recorda quina contrasenya s'havia utilitzat, en Olvide la contrasenya et permet canviar-la directament.

Un cop omplert el usuari i contrasenya s'ha de seleccionar en Entrar situat a la pantalla superior esquerra.

A1.11. Registre

Registro: pantalla utilitzada per registrar a l'usuari en l'aplicació.

Figura 64: Registre

Per registrar-se en l'aplicació es necessiten omplir les dades indicades en la pantalla:
Usuario→ s'ha d'indicar el email amb el que es vol Identificar a l'aplicació.
Nombre → s'ha de indicar el nom de l'usuari que es vol Identificar.
Contraseña → indicar la contrasenya que vol tenir
Repite la contraseña → repetir la contrasenya per comprovar que s'ha escrit correctament.

A1.12. Comanda

Pedido  en aquesta pantalla es pot veure que s'ha anat afegint a la comanda abans d'enviar-la a la botiga. En aquesta pantalla es pot veure el mail, nom d'usuari i la data i la llista de plats que composen la comanda. En la llista de plats podem veure el nom del plat i a sota el numero que apareix és la quantitat que es demana en la comanda.

Els iconos del  i  tenen el mateix funcionament que en la llista de plats.
Un cop comprobada la comanda podeu seleccionar sobre Enviar Pedido per enviar la comanda a la botiga .

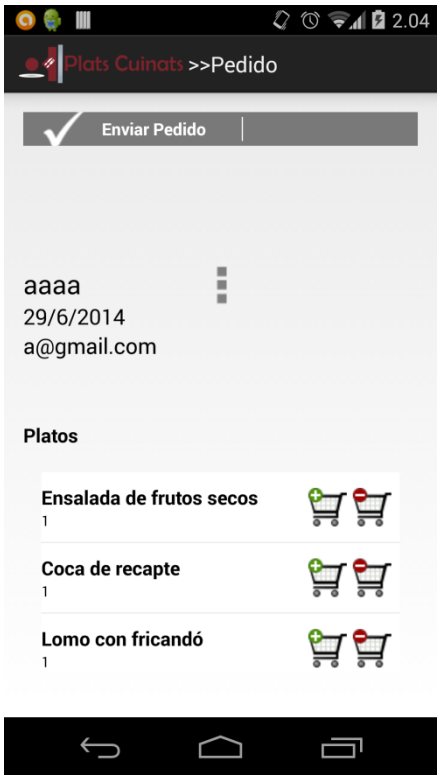

Figura 65: Comanda

Des d'aquesta pantalla es pot afegir una nou plat a la comanda seleccionant la icona dels 3 punts i apareixerà un nou submenú, des d'on es pot anar a la pantalla on hi ha la llista de plats i afegir el que es vulgui.

Quan la comanda estigui llesta s'ha d'enviar a la botiga seleccionant el botó Enviar

Figura 66: Afegir plat a la comanda

Des del menú Logout també es pot sortir de l'aplicació.

Annex 2: Instal·lació ADT

Per instal·lar el plugin:

1) Iniciem Eclipse i anem a Help/Install New Software

2) Cliquem Add en la part superior dreta de la finestra que s'obre

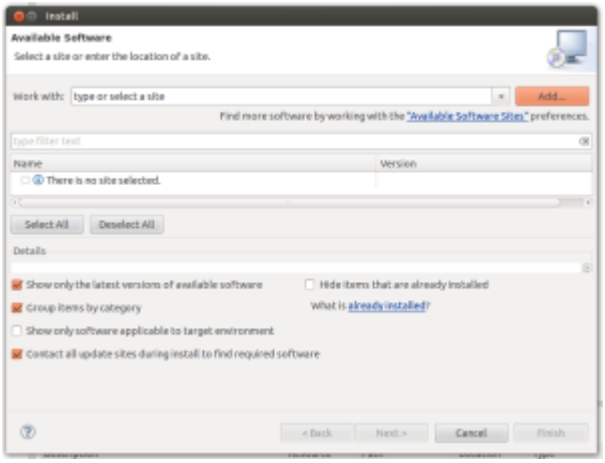

Figura 67: Instal·lació ADT

3) Al pitjar Add s'obrirà una nova finestra on s'ha de posar ADT Plugin en el nom i en localització la url: https://dl-ssl.google.com/android/eclipse/

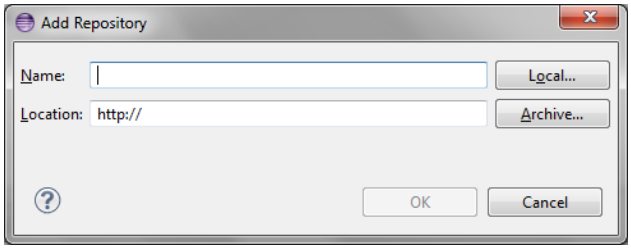

Figura 68: Add finestra

4) Fem Ok

5) Selecciones Developer Tools i fem següent

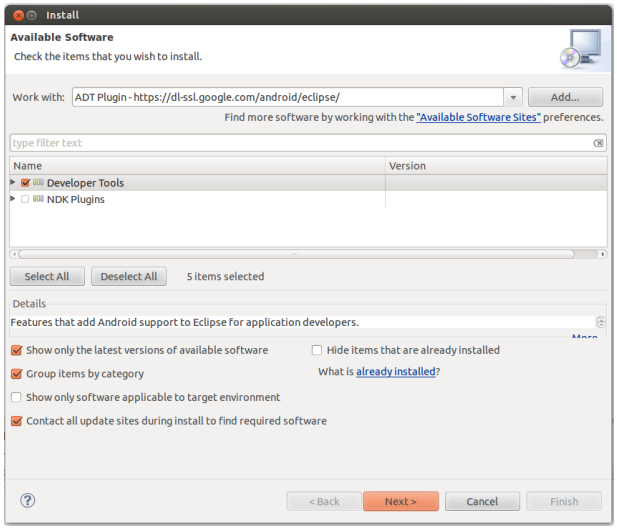

Figura 69: Developers Tools

6) En la següent pantalla es veuen les eines que s'instal·laran i fem següent

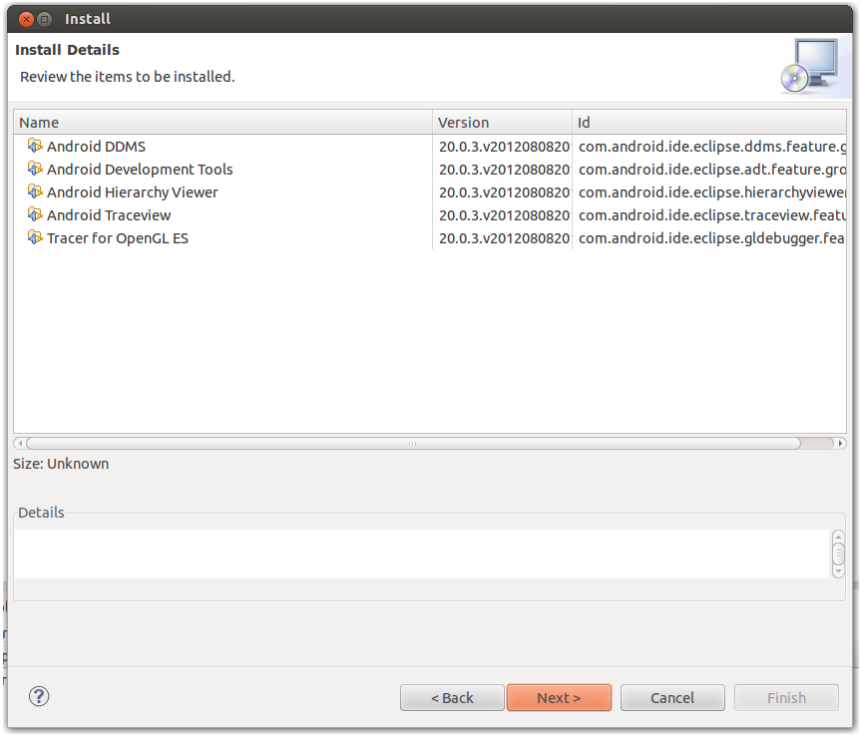

Figura 70: Paquets a instal·lar

7) En la següent pantalla s'han d'anar acceptant les llicencies i acabem la instal·lació. Quan estigui instal·lat reiniciem l'Eclipse i ja podem treballar.

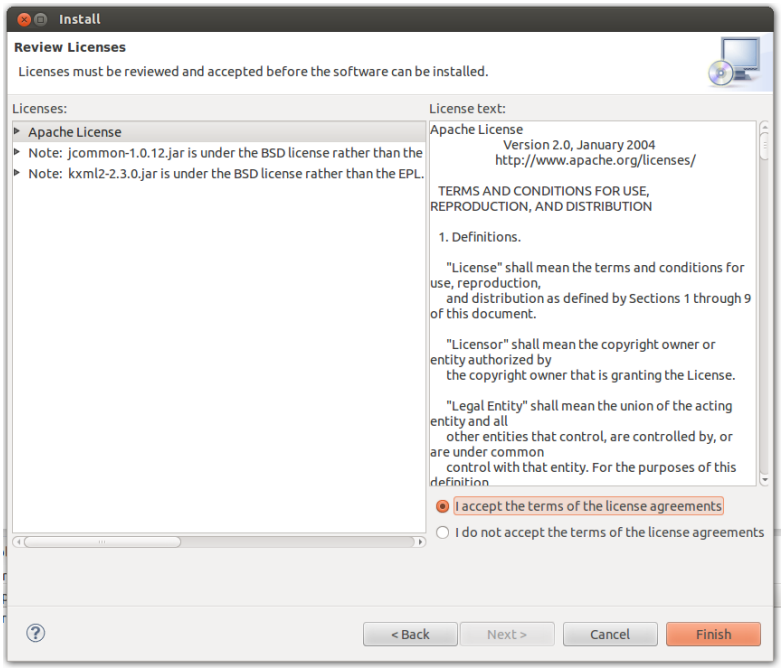

Figura 71: Acceptació de llicències

Annex 3: API Google Maps

A3.1. Instal·lar la llibreria de Google Play Services

Instal·lar la llibreria que inclou API Google Maps v2. Per instal·lar la llibreria s'ha de descarregar mitjançant el SDK Manager de Android, en la secció de Extras seleccionar Google Play Services.

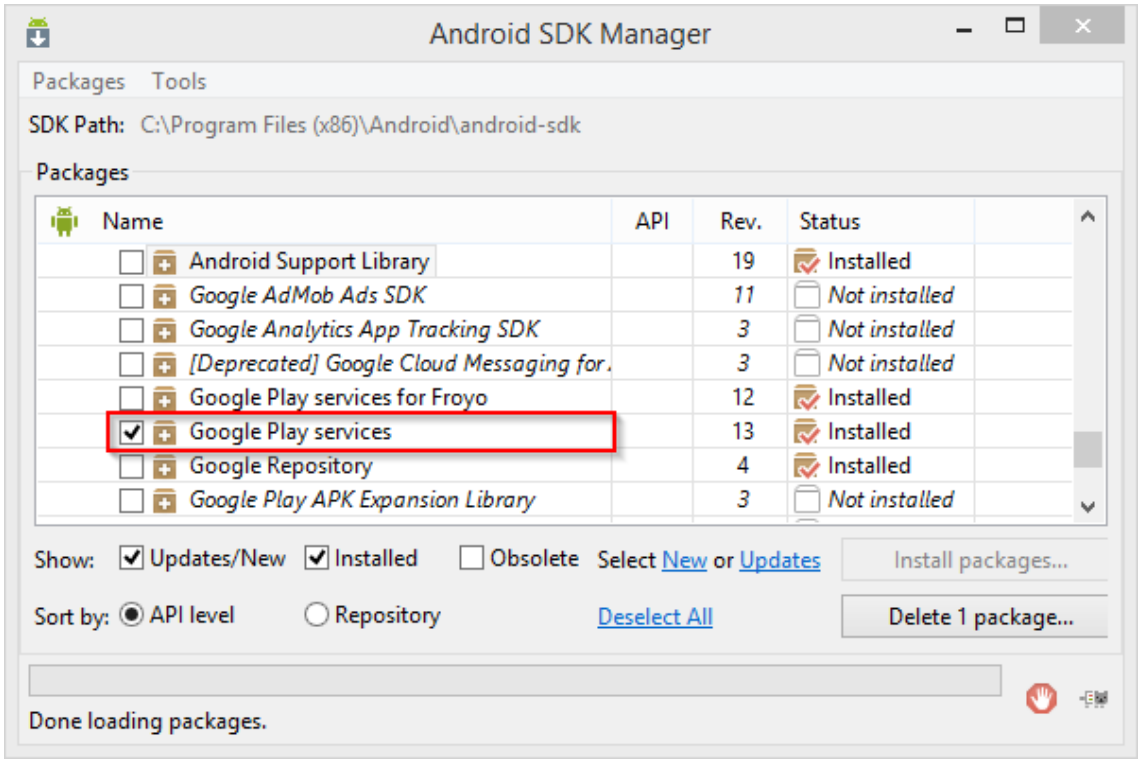

Figura 72: Google Play Services

Un cop descarregat es pot trobar en la ruta *<ruta-sdk>\extras\google\google_play_services\libproject\google-play-services_lib* i només caldrà importar el codi al nostre projecte.

Per importar hem d'anar al menú "*File/Import* " i seleccionar la opció "*Android/Existing Android Code Into Workspace*"

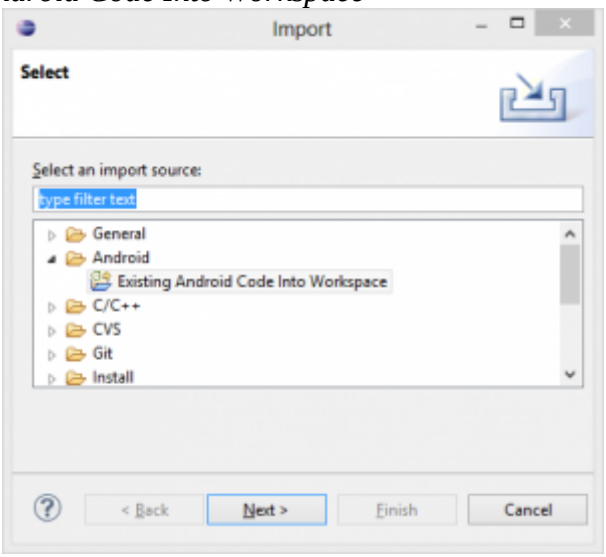

Figura 74: Importar Google Play Services

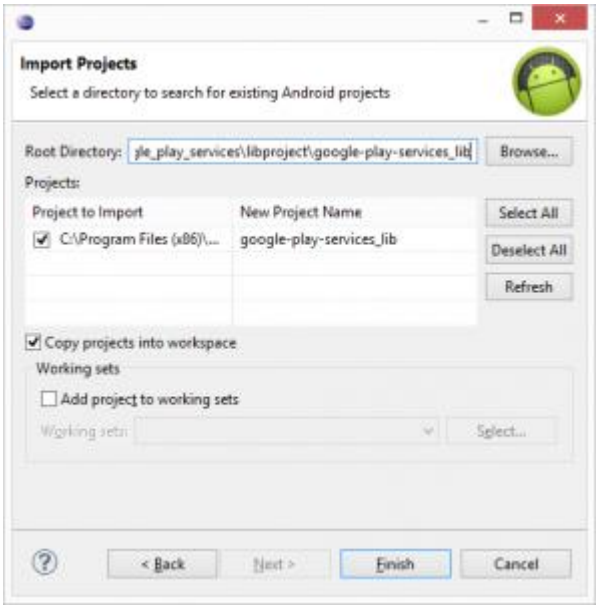

Figura 73: Importar Google Play Services (II)

Seleccionem el botó "*Next*" i accedim a la pantalla amb les opcions de importació. En el camp "*Root Directory*" indiquem la ruta on es troba el projecte i selecciones el projecte anomenat *google-play-services-lib* en la llista de "*Projects to import*" i marquem la opció "*Copy projects into workspace*"

Fen Finish ja hem importat el projecte. Per últim hem de veure dins de les propietats del projecte (boto dret propietats) i assegurar-nos que esta marcat dins la secció "Java Build Path" la opció "Android Private Libraries".

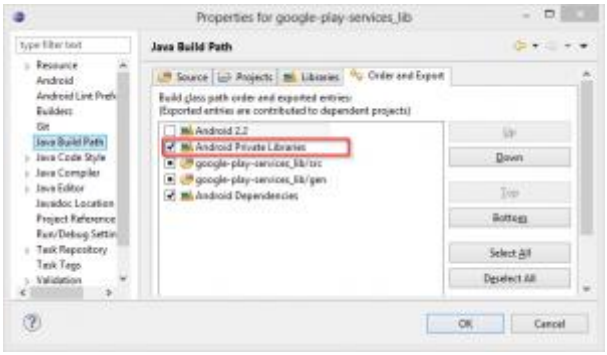

Figura 75: Androdi Private Libraries

A3.2. Crear un projecte que contingui la llibreria de mapes

Per utilitzar aquesta llibreria en el nostre projecte el que s'ha de fer és accedir a les propietats del projecte on afegir la llibreria i dins de l'apartat "Android" s'ha de afegir en l'apartat "Library" aquesta llibreria.

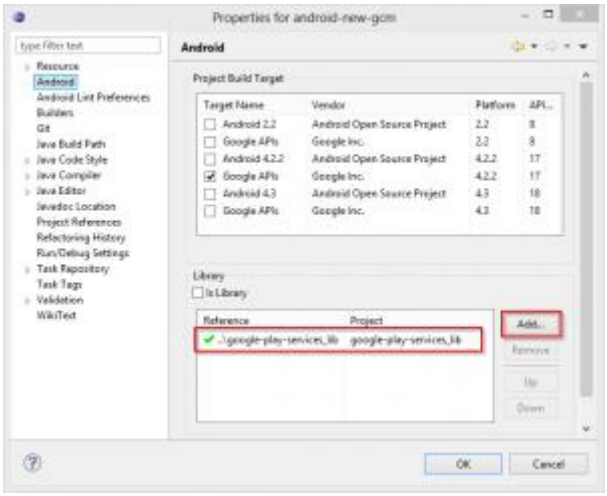

Figura 76: Llibreria a incloure al projecte

A3.3. Afegir el permís d'ús de la llibreria dins el AndroidManifest.xml

Afegir en el AndroidManifest.xml s'ha d'afegir la clàusula <meta-data> dins de l'element <aplication>

```
<meta-data android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version" />
```

I finalment dins de fitxer proguard-project.txt les següents línees per evitar que aquesta eina elimini algunes classes necessàries.

```
-keep class * extends java.util.ListResourceBundle {
  protected Object[][] getContents();
}

-keep public class
com.google.android.gms.common.internal.safeparcel.SafeParcelable {
  public static final *** NULL;
}

-keepnames @com.google.android.gms.common.annotation.KeepName class *
-keepclassmembernames class * {
  @ccom.google.android.gms.common.annotation.KeepName *;
}

-keepnames class * implements android.os.Parcelable {
  public static final ** CREATOR;
}
```

A3.4. Aconseguir una Map API Key per poder mostrar mapes en l'aplicació

Primer hem d'accedir a la Consola de APIs de Google. Un cop a dins s'ha de crear un nou projecte.

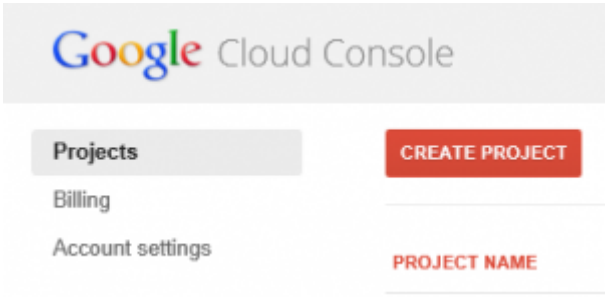

Figura 77: Crear projecte Google Cloud Console

Al crear el nou projecte es demanarà el nom del projecte per assignar-li un identificador únic.

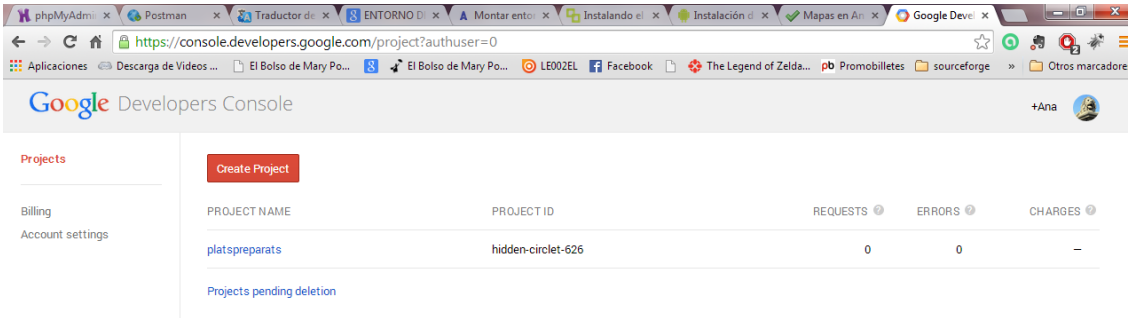

Figura 78: Crear nou projecte

Un cop el projecte està creat s'ha d'accedir a l'opció "APIs &Auth" on es poden activar i desactivar cadascuna de les APIs de Google que volem utilitzar.

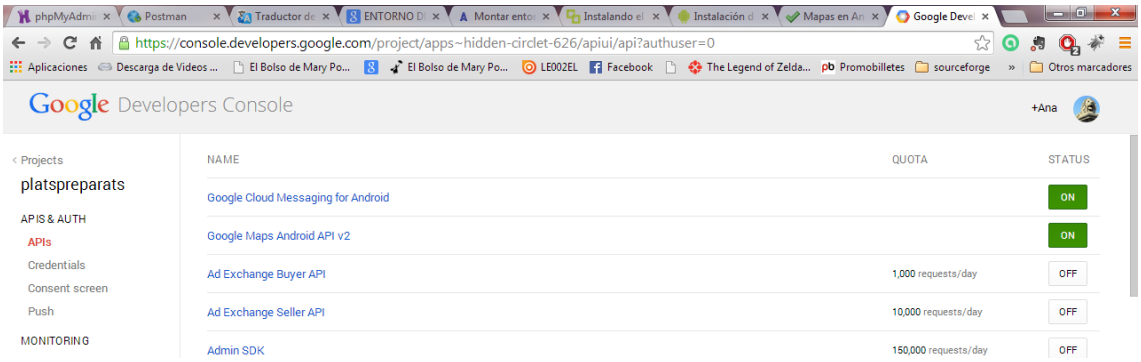

Figura 79: Activació APIs

"*Registered Apps*" del menú esquerra (submenú "*APIs &Auth*" seleccionarem el botó "*REGISTER APP*" per registrar l'aplicació.

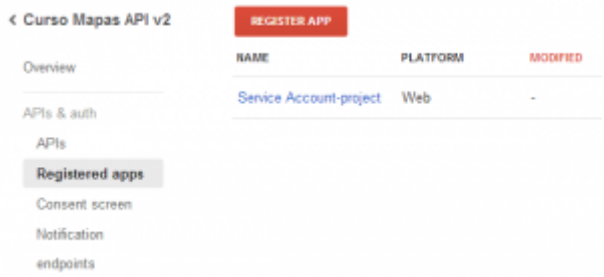

Figura 80: Registrar aplicació

Accedint a aquesta opció podem obtenir la *API Key* que ens permeti utilitzar el servei de mapes des de la nostra aplicació particular. Tenir que indicar el nom de l'aplicació, el tipus (*Android*), el mode d'accés al API (en el nostre cas directe des d'*Android*), el paquet *Java* utilitzat (*com.pfc.platspreparats*) i la empremta digital *SHA1* del certificat amb el que signem la nostra aplicació.

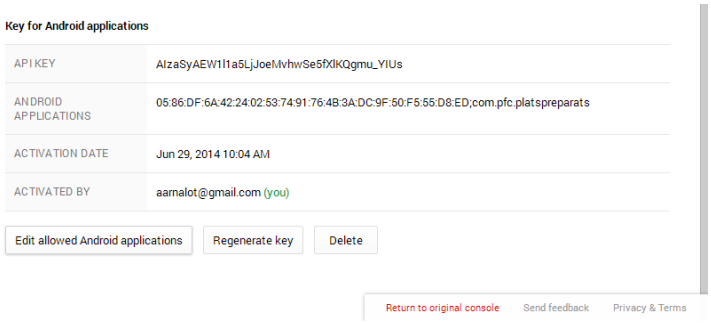

Figura 81: Registre aplicació Google

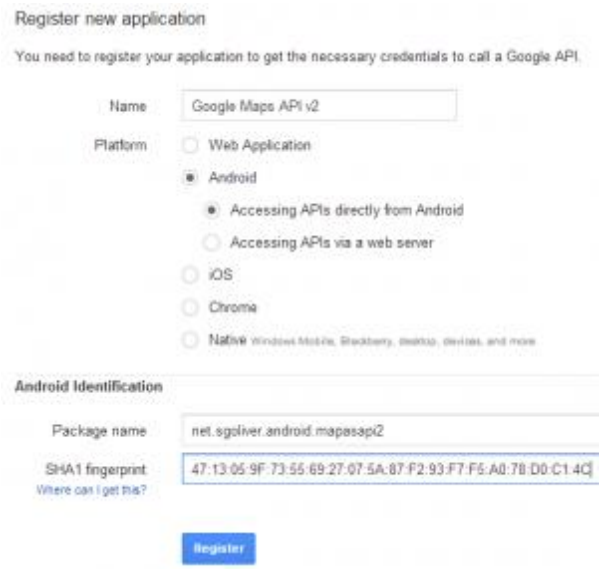

Figura 82: Registre real aplicació a Google

Per aconseguir "*SHA1 fingerprint*" que es demana en el registre s'ha d'anar a la configuració d'*Eclipse* i dins de el menú *Window/Preferences* anant a la secció *Android/Built* podem consultar-ho per introduir-lo en la consola de *Google*.

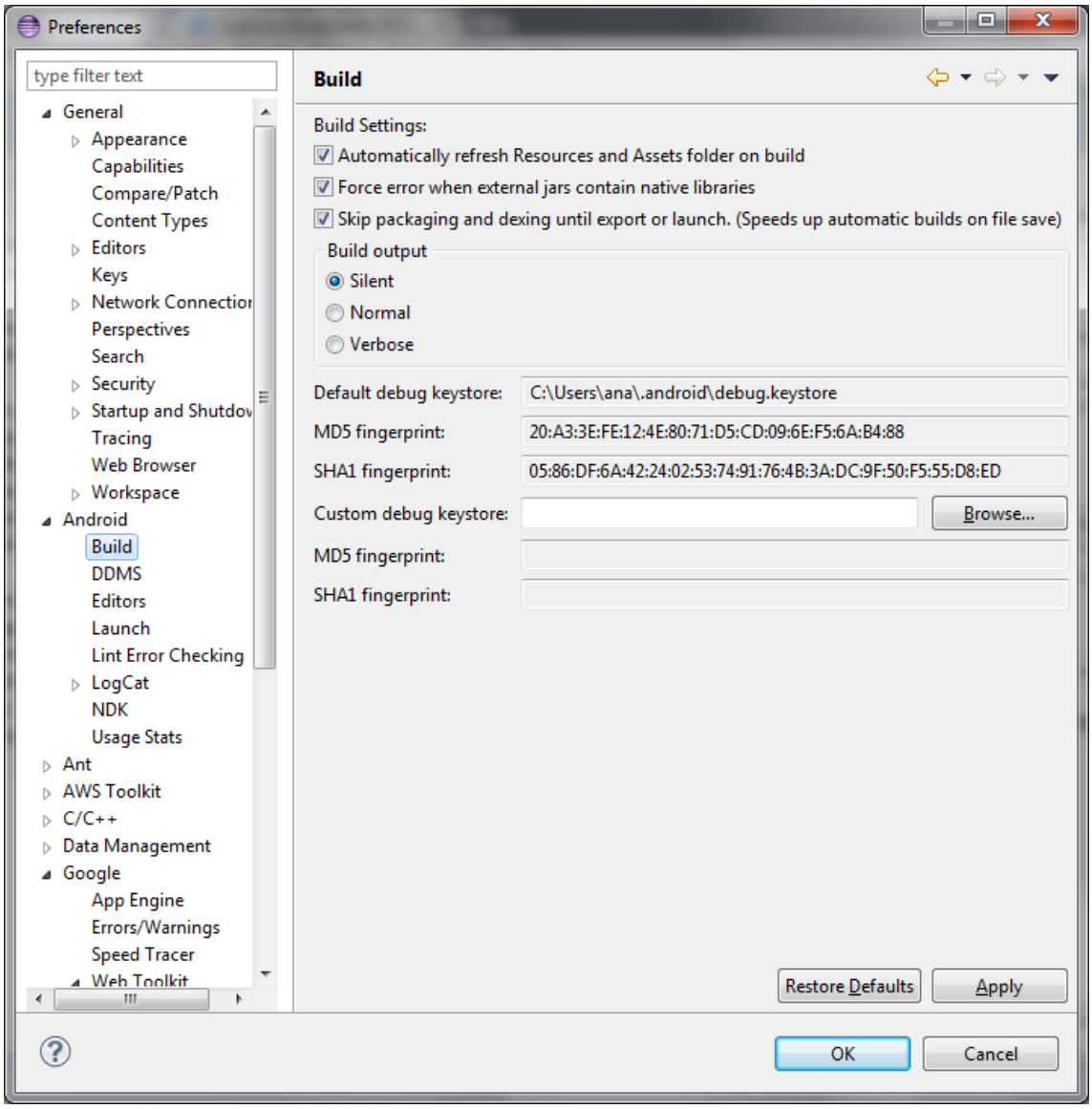

Figura 83: SHA Fingerprint

Un cop omplert tot el que ens demana fem registrar i podrem obtenir la nostra API Key per incorporar-la a l'aplicació.

Annex 4: Diagrama de classes detallat

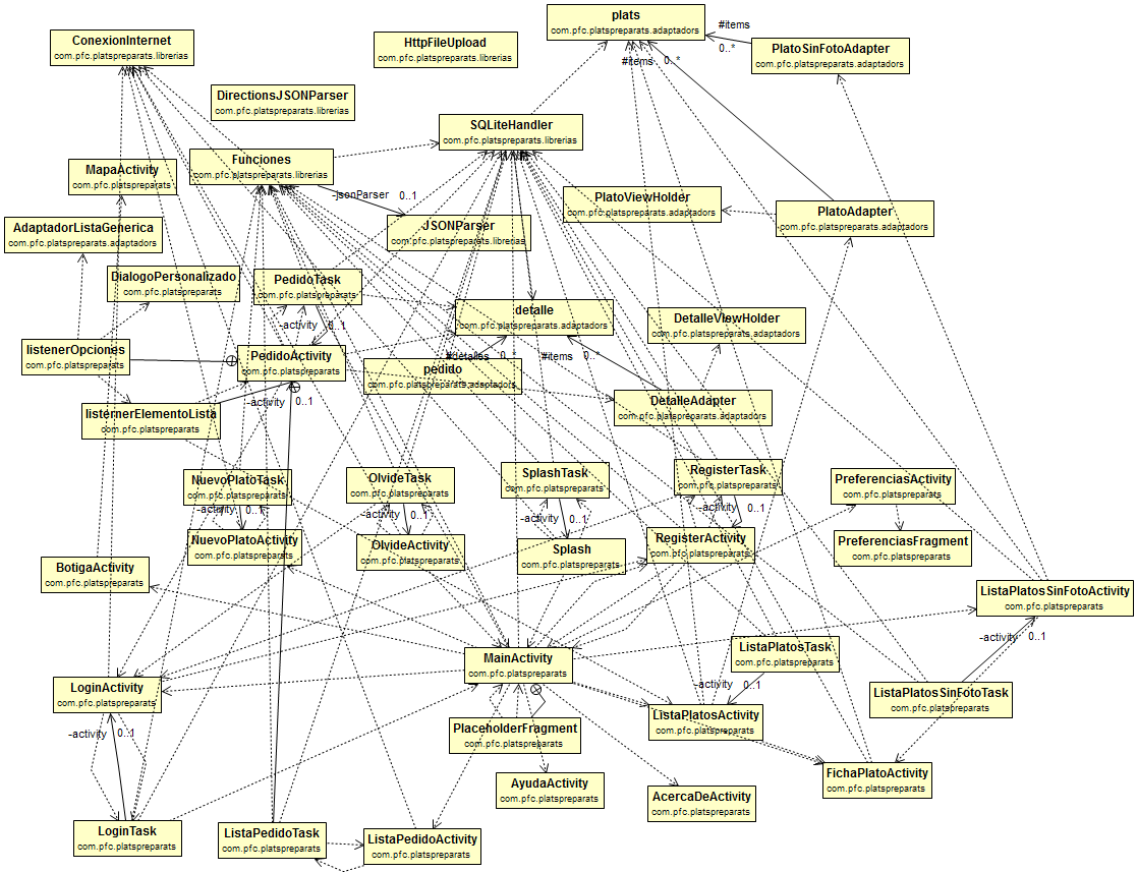

Figura 84: Diagrama de classes

- *com.pfc.platspreparats.librerias*

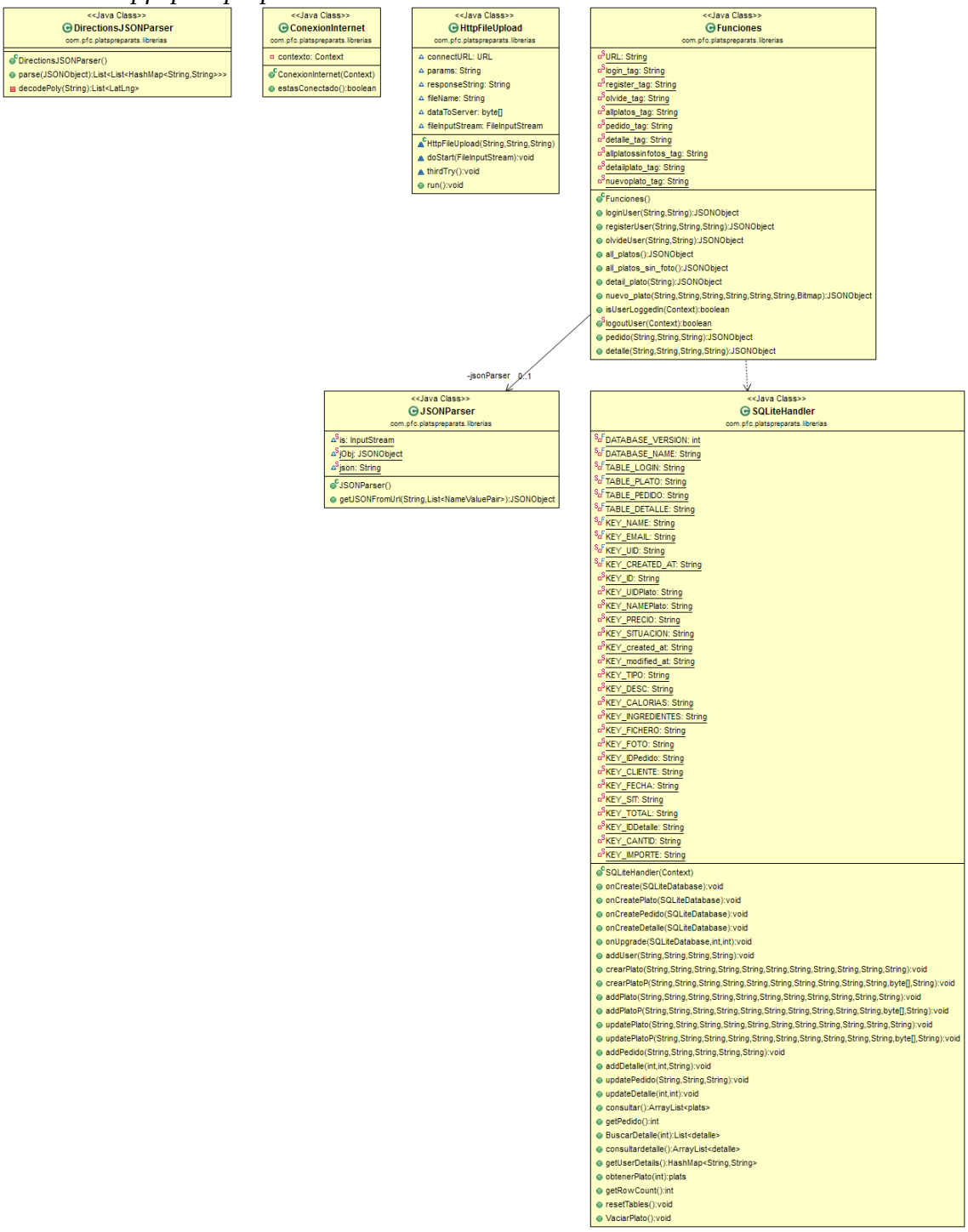

Figura 85: Classes paquet librerias

- *com.pfc.platspreparats.adaptadors*

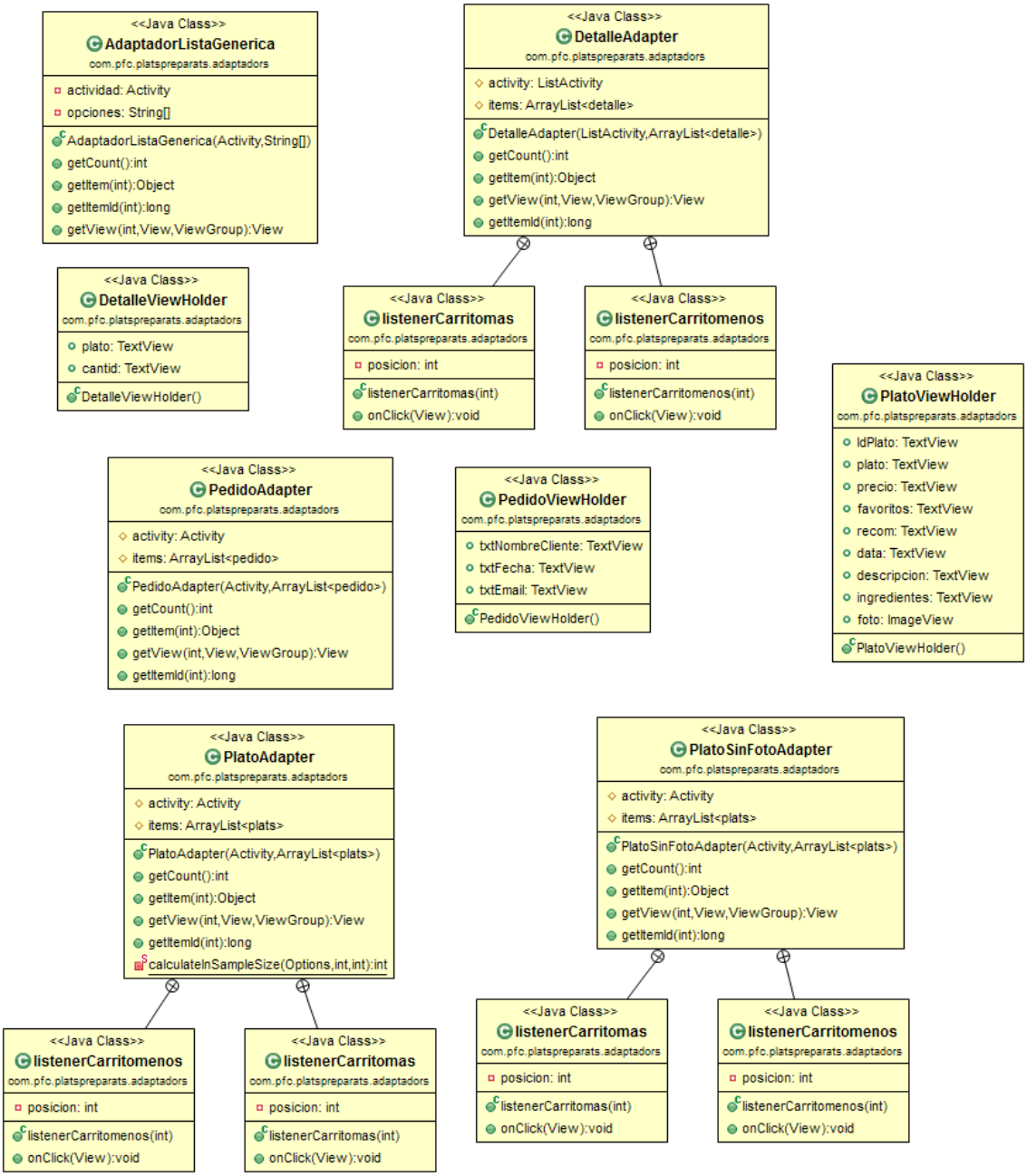

Figura 86: Diagrama Classes Adaptadores

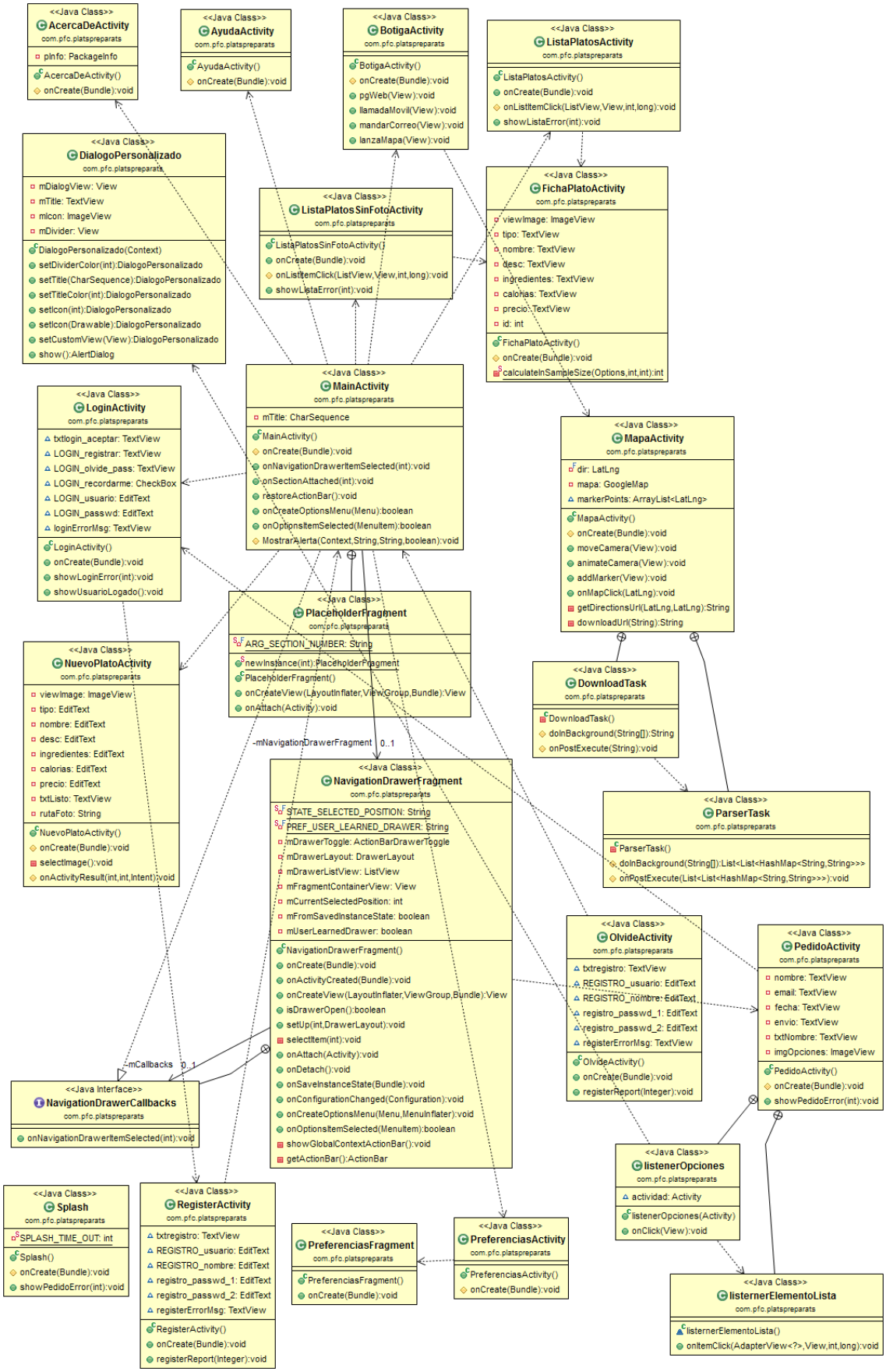

Figura 87: Diagrama de clases Activitys

Annex 5: Especificació Alt Nivell de la resta de casos d'ús

A5.1. Usuari

Taula 9: Configurar

Nom	Configurar
Descripció	Configuració aplicació
Actors	Usuari
Pre-condicions	Cap
Flux normal	1.Seleccionar sobre el menú de configuració 2.Seleccionar entre les opcions disponibles
Excepcions	Cap
Postcondicions	Es pot seleccionar quin tipus d'acció vol que faci el telèfon al rebre una notificació i aquesta es guardarà en les preferències de l'aplicació per tenir-ho disponible sempre.

Taula 10: Consultar Acerca De

Nom	Consultar Acerca De
Descripció	Consulta dades de l'aplicació
Actors	Usuari
Pre-condicions	Cap
Flux normal	1.Seleccionar el menú Acerca De de l'aplicació
Excepcions	Cap
Postcondicions	Cap

Taula 11: Consultar Ajuda

Nom	Consultar Ajuda
Descripció	Consulta Ajuda de l'aplicació
Actors	Usuari
Pre-condicions	Cap
Flux normal	1.Seleccionar el menú Ajuda de l'aplicació
Excepcions	Cap
Postcondicions	Cap

Taula 12: Consultar Botiga

Nom	Consultar Botiga
Descripció	Consulta dades de la botiga
Actors	Usuari
Pre-condicions	Cap
Flux normal	1.Seleccionar el menú Tienda de l'aplicació
Excepcions	Cap
Postcondicions	Cap

Taula 13: Consultar Mapa

Nom	Consultar Mapa
Descripció	Consulta localització botiga en un mapa
Actors	Usuari
Pre-condicions	Per tenir la localització del usuari s'ha de tenir la localització activada al telèfon i estar en la pantalla botiga
Flux normal	1.Seleccionar sobre l'adreça de la botiga 2.Apareix un mapa amb una icona de la botiga en el mapa 3.Simou el mapa per tornar a la posició de la botiga s'ha de seleccionar sobre el botó Tienda. 4.Si es vol mostrar la localització del usuari i la ruta fins a la botiga s'ha de seleccionar el botó Yo. 5.Amb el botó Marcador es pot marcar un punt concret del mapa.

Taula 14: Telefonar

Nom	Telefonar
Descripció	Trucar per telèfon a la botiga
Actors	Usuari
Pre-condicions	Estar en la pantalla de la botiga
Flux normal	1.Seleccionar sobre el telèfon de la botiga, això permet trucar directament des del mòbil a la botiga
Excepcions	Cap
Postcondicions	Cap

Taula 15: Enviar Email

Nom	Enviar Email
Descripció	Enviar email al mail de la botiga
Actors	Usuari
Pre-condicions	Estar en la pantalla botiga
Flux normal	1.Seleccionar sobre l'adreça de mail de la pantalla 2. Obrirà un quadre de diàleg per seleccionar el procés a través del que es vol enviar el mail 3.Escriure mail i enviar-lo a través de l'aplicació seleccionada.
Excepcions	Cap
Postcondicions	Cap

A5.2. Usuari registrat

Taula 16: Sortir

Nom	Sortir
Descripció	DesIdentificar-se de l'aplicació
Actors	Usuari registrat
Pre-condicions	Estar Identificat
Flux normal	1.Seleccionar menú logout
Excepcions	Cap
Postcondicions	Esborrar l'usuari de la base de dades del telèfon on es mantenia connectat.